```
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSSS  YYY         YYY
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSSS  YYY         YYY
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSSS  YYY         YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS              YYY         YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS              YYY         YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS              YYY         YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS                 YYY   YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS                 YYY   YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS                 YYY   YYY
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP        SSSSSSSSS           YYY
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP        SSSSSSSSS           YYY
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP        SSSSSSSSS           YYY
UUU         UUU  EEE                    TTT        PPP                          SSS        YYY
UUU         UUU  EEE                    TTT        PPP                          SSS        YYY
UUU         UUU  EEE                    TTT        PPP                          SSS        YYY
UUU         UUU  EEE                    TTT        PPP                          SSS        YYY
UUU         UUU  EEE                    TTT        PPP                          SSS        YYY
UUU         UUU  EEE                    TTT        PPP                          SSS        YYY
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP              SSSSSSSSSSSS           YYY
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP              SSSSSSSSSSSS           YYY
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP              SSSSSSSSSSSS           YYY
```

```
_$


Va
--
00
00
00
00
00
48
7F
7F
7F
7F
7F
7F
7F
7F
7F
7F
7F
7F
7F
7F
7F
7F
```

```
**FILE**ID**UETCLIGOO                         I  6

UU      UU EEEEEEEEEE TTTTTTTTTT   CCCCCCCC LL         IIIIII     GGGGGGGG   OOOOOO      OOOOOO
UU      UU EEEEEEEEEE TTTTTTTTTT   CCCCCCCC LL         IIIIII     GGGGGGGG   OOOOOO      OOOOOO
UU      UU EE             TT     CC         LL           II     GG         OO      OO  OO      OO
UU      UU EE             TT     CC         LL           II     GG         OO      OO  OO      OO
UU      UU EE             TT     CC         LL           II     GG         OO    OOOO  OO    OOOO
UU      UU EE             TT     CC         LL           II     GG         OO    OOOO  OO    OOOO
UU      UU EEEEEEEE       TT     CC         LL           II     GG         OO  OO  OO  OO  OO  OO
UU      UU EEEEEEEE       TT     CC         LL           II     GG         OO  OO  OO  OO  OO  OO
UU      UU EE             TT     CC         LL           II     GG  GGGGGG OOOO    OO  OOOO    OO
UU      UU EE             TT     CC         LL           II     GG  GGGGGG OOOO    OO  OOOO    OO
UU      UU EE             TT     CC         LL           II     GG      GG OO      OO  OO      OO   ....
UU      UU EE             TT     CC         LL           II     GG      GG OO      OO  OO      OO   ....
UUUUUUUUUU EEEEEEEEEE     TT       CCCCCCCC LLLLLLLLLL IIIIII     GGGGGG     OOOOOO      OOOOOO     ....
UUUUUUUUUU EEEEEEEEEE     TT       CCCCCCCC LLLLLLLLLL IIIIII     GGGGGG     OOOOOO      OOOOOO     ....

LL         IIIIII     SSSSSSSS
LL         IIIIII     SSSSSSSS
LL           II     SS
LL           II     SS
LL           II     SS
LL           II     SS
LL           II       SSSSSS
LL           II       SSSSSS
LL           II             SS
LL           II             SS
LL           II             SS
LL           II             SS
LLLLLLLLLL IIIIII   SSSSSSSS
LLLLLLLLLL IIIIII   SSSSSSSS
```

```
UE
VO

65
59
45
20

20
65
61
20
2E

20
74
73
2E

20
64
73
53

20
72
20
63

20
75
72
41

20
21
61
6F

20
21

20
```

Table of contents

```
0000            .TITLE UETCLIG00 VAX/VMS UETP Cluster Integration Test
0000            .IDENT  'V04-000'
0000            .ENABLE SUPPRESSION
0000
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000
0000    ;++
0000    ; FACILITY:
0000    ;       This module will be distributed with VAX/VMS under the [SYSTEST]
0000    ;       account.
0000    ;
0000    ; ABSTRACT:
0000    ;       This module is the Cluster Integration phase of the UETP.  It tests
0000    ;       that the node from which it is run fits in with all other nodes in
0000    ;       a cluster, trying those basic functions of a cluster which are
0000    ;       accessible to typical user programs.
0000    ;
0000    ; ENVIRONMENT:
0000    ;       Because of the requirement that all error messages be displayed at
0000    ;       the terminal that is running the UETP, all errors reported by a slave
0000    ;       process must be sent to the master process.  We have chosen to do that
0000    ;       by copying (via $PUTMSG action routine) slave messages of other than
0000    ;       success severity to a disk file, and then relaying that file to the
0000    ;       master process at the end of the test.  The file, SYS$ERROR.LOG,
0000    ;       should be automatically deleted at the end of the test.
0000    ;
0000    ;       Note that the test assumes that DECnet node names correspond to cluster
0000    ;       node names!
0000    ;
0000    ;       This program will run in user access mode except when getting a copy
0000    ;       of VMS's configuration data base.  We require the following
0000    ;       privileges and quotas:
0000    ;               CMKRNL
0000    ;
0000    ;--
0000    ;
```

```
0000    ; AUTHOR: Richard Holstein,      CREATION DATE: June, 1983
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;         V03-009 RNH0008          Richard N. Holstein,    05-Jul-1984
0000    ;                 Fix Spelling error in message, add message to warn if deadlock
0000    ;                 detection is turned off.
0000    ;
0000    ;         V03-008 RNH0007          Richard N. Holstein,    29-Apr-1984
0000    ;                 Have SCSNODE return the entire string, not just 4 chars.  Have
0000    ;                         NO_NODE_MSG be a warning, not info message.
0000    ;
0000    ;         V03-007 WHM0001          Bill Matthews           14-Apr-1984
0000    ;                 Replace reference to SCSNODEL and SCSNODEH with SCSNODE.
0000    ;
0000    ;         V03-006 RNH0006          Richard N. Holstein,    11-Apr-1984
0000    ;                 Use correct error message if a node has no disk DDBs for file
0000    ;                 test.  Allow multiple strings to be encoded in the MODE logical
0000    ;                 name.  Test blocking ASTs in a cluster and allow the test to
0000    ;                 $HIBER minimally or not at all if deadlock detection is quick.
0000    ;
0000    ;         V03-005 RNH0005          Richard N. Holstein,    24-Feb-1984
0000    ;                 Fix SSERROR interaction with RMS_ERROR.  Change sentinel lines
0000    ;                 from slave process log files so that we may copy them into the
0000    ;                 master log without the test controller thinking that they are
0000    ;                 sentinels from the master process.  Indent all of slave log
0000    ;                 file lines copied, including embedded newlines.
0000    ;
0000    ;         V03-004 RNH0004          Richard N. Holstein,    07-Jan-1984
0000    ;                 Be more choosey about the nodes we'll allow for lock testing
0000    ;                 and for file testing:  ensure that we believe a VMS node is a
0000    ;                 member of our cluster and that the path to all nodes is in
0000    ;                 good shape.
0000    ;
0000    ;         V03-003 RNH0003          Richard N. Holstein,    22-Nov-1983
0000    ;                 Fix params to DEADLOCK_WAIT error message.
0000    ;
0000    ;         V03-002 RNH0002          Richard N. Holstein,    26-Sep-1983
0000    ;                 Fix RET from subroutine which should be RSB.  Change trace
0000    ;                 logical name to MODE to avoid naming conflict and be compatible
0000    ;                 with the rest of UETP.  Add SE_NAM so correct SYS$ERROR.LOG file
0000    ;                 is always $ERASEd.
0000    ;
0000    ;         V03-001 RNH0001          Richard N. Holstein,    28-Jul-1983
0000    ;                 Add shared file access, new UETP messages and file access
0000    ;                 debugging info.
0000    ;
0000    ;**
```

```
0000          .SBTTL  Declarations
0000  ;
0000  ; INCLUDE FILES:
0000  ;
0000  ;       SYS$LIBRARY:LIB.MLB     for general definitions
0000  ;       SHRLIB$:UETP.MLB        for UETP definitions
0000
0000  ;
0000  ; MACROS:
0000  ;
0000          $CHFDEF                         ; Condition handler frame definitions
0000          $BRKDEF                         ; $BRKTHRU flags
0000          $DVIDEF                         ; $GETDVI ITMLST item codes
0000          $IODEF                          ; I/O function codes
0000          $JPIDEF                         ; $GETJPI ITMLST item codes
0000          $LCKDEF                         ; $ENQ flags and miscellany
0000          $NAMDEF                         ; NAM block definitions and constants
0000          $PBDEF                          ; Path block definitions
0000          $SHRDEF                         ; Shared messages
0000          $STSDEF                         ; Status return
0000          $SYIDEF                         ; $GETSYI ITMLST item codes
0000          $UETIDBDEF                      ; UETP I/O database definitions
0000          $UETPDEF                        ; UETP
0000
0000  .MACRO  MESSAGES                        ; Define msgs between master and slaves
0000          DEFMSG  HELLO                   ; Identify master to slave
0000          DEFMSG  IMOK                    ; Slave got correctly set up
0000          DEFMSG  TAKELOCK                ; Tell slave to take out a lock
0000          DEFMSG  GOTLOCK                 ; Slave successfully took out a lock
0000          DEFMSG  QUEUELOCK               ; Slave is queued for a lock (deadlock)
0000          DEFMSG  DEADLOCK                ; Slave was chosen as a deadlock victim
0000          DEFMSG  ACCESS                  ; Tell slave to access a file
0000          DEFMSG  CONTINUE                ; Slave is accessing a file
0000          DEFMSG  MOVE_ON                 ; Section finished, continue with next
0000          DEFMSG  ERRORLOG                ; Slave is sending a copy of SYS$ERROR
0000          DEFMSG  ERRORLOG_ENDED          ; Slave is finished sending SYS$ERROR
0000  .ENDM   MESSAGES
0000
0000  .MACRO  BEQLW   DISPL,?L1               ; Word displacement branch if equal
0000          BNEQ    L1                      ; Reverse the sense of the test...
0000          BRW     DISPL                   ; ...so that the false passes over
0000  L1:
0000  .ENDM   BEQLW
0000
0000  .MACRO  BNEQW   DISPL,?L1               ; Word displacement branch if not equal
0000          BEQL    L1                      ; Reverse the sense of the test...
0000          BRW     DISPL                   ; ...so that the false passes over
0000  L1:
0000  .ENDM   BNEQW
0000
0000  .MACRO  BLBCW   SRC,DISPL,?L1           ; Word displacement BR on low bit clear
0000          BLBS    SRC,L1                  ; Reverse the sense of the test...
0000          BRW     DISPL                   ; ...so that the false passes over
0000  L1:
0000  .ENDM   BLBCW
0000
0000  .MACRO  BLBSW   SRC,DISPL,?L1           ; Word displacement BR on low bit set
```

```
          0000                  BLBC    SRC,L1                  ; Reverse the sense of the test...
          0000                  BRW     DISPL                   ; ...so that the false passes over
          0000          L1:
          0000          .ENDM   BLBSW
          0000
          0000          .MACRO  BBCW    POS,BASE,DISPL,?L1      ; Word displacement BR on bit clear
          0000                  BBS     POS,BASE,L1             ; Reverse the sense of the test...
          0000                  BRW     DISPL                   ; ...so that the false passes over
          0000          L1:
          0000          .ENDM   BBCW
          0000
          0000          .MACRO  BBSW    POS,BASE,DISPL,?L1      ; Word displacement BR on bit set
          0000                  BBC     POS,BASE,L1             ; Reverse the sense of the test...
          0000                  BRW     DISPL                   ; ...so that the false passes over
          0000          L1:
          0000          .ENDM   BBSW
          0000
          0000          ;
          0000          ; EQUATED SYMBOLS:
          0000          ;
          0000          ;       Facility number definitions:
00000001  0000                  RMS$_FACILITY = 1
          0000
          0000          ;       SHR message definitions:
00740000  0000                  UETP = UETP$_FACILITY@STS$V_FAC_NO ; Define the UETP facility code
007410E0  0000                  UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
00741038  0000                  UETP$_BEGIND = UETP!SHR$_BEGIND
00741080  0000                  UETP$_ENDEDD = UETP!SHR$_ENDEDD
00741130  0000                  UETP$_TEXT   = UETP!SHR$_TEXT
          0000
          0000          ;       Internal flag bits...:
00000001  0000                  CLIG_V_DEADNODE = 1             ; Marks a slave node as out of the test
          0000                                                  ; Kept in one of NODE_NAMES descriptors
00000000  0000                  CLIG_V_DEBUG    = 0             ; Remembers if running in debug mode
          0000                                                  ; Kept in FLAGS
00000001  0000                  CLIG_V_SLAVE    = 1             ; Remembers if I'm a slave or a master
          0000                                                  ; Kept in FLAGS
00000002  0000                  CLIG_V_SE_DEAD  = 2             ; Set if can't write to SYS$ERROR.LOG
          0000                                                  ; Kept in FLAGS
00000003  0000                  CLIG_V_BEGINMSG = 3             ; Set if we've typed beginning message
          0000                                                  ; Kept in FLAGS
          0000          ;       ...and corresponding masks:
00000002  0000                  CLIG_M_DEADNODE = 1@CLIG_V_DEADNODE
00000001  0000                  CLIG_M_DEBUG    = 1@CLIG_V_DEBUG
00000002  0000                  CLIG_M_SLAVE    = 1@CLIG_V_SLAVE
00000004  0000                  CLIG_M_SE_DEAD  = 1@CLIG_V_SE_DEAD
00000008  0000                  CLIG_M_BEGINMSG = 1@CLIG_V_BEGINMSG
          0000
          0000          ;       Miscellany:
          0000                  .MACRO  DEFMSG  MSGNAM          ; Compute the longest message name
          0000                  MSGNAM'_LENGTH = %LENGTH(MSGNAM)
          0000                  .IIF LT MAX_MSGNAM_LENGTH - MSGNAM'_LENGTH,-
          0000                          MAX_MSGNAM_LENGTH = MSGNAM'_LENGTH
          0000                  .ENDM   DEFMSG
00000000  0000                  MAX_MSGNAM_LENGTH = 0           ; Set up an initial value
          0000                  MESSAGES                        ; Set up MAX_MSGNAM_LENGTH final value
000000C8  0000                  TEXTB_SIZE      = 200           ; Internal text buffer size
```

```
                      0000                                         ; Also, maximum length of msg to send
                      0000                                         ; We must pass a filespec as a mesasge
                      0000       .IIF LT TEXTB_SIZE - NAM$C_MAXRSS - MAX_MSGNAM_LENGTH,-
            0000010D  0000              TEXTB_SIZE = NAM$C_MAXRSS + MAX_MSGNAM_LENGTH
            00000001  0000       SS_SYNCH_EFN    = 1               ; EFN for synchronizing system svcs
            000000FF  0000       MAX_NODES       = 255             ; Maximum number of nodes per cluster
            0000000F  0000       PRCNAM_LENGTH   = 15              ; Maximum length of a process name
            00000006  0000       NODE_LENGTH     = 6               ; Maximum length of a node name
            00000005  0000       UNIT_LENGTH     = 5               ; Maximum length of a device unit number
            0000005A  0000       PATTERN_1       = ^X5A            ; Data pattern for test files 1st block
            000000F0  0000       PATTERN_2       = ^XF0            ; Data pattern for test files 2nd block
            0000003C  0000       BRKTHRU_TIMOUT  = 60              ; Seconds to wait for cluster $BRKTHRU
            0000003C  0000       QIO_TIMEOUT     = 60              ; Seconds to wait for DECnet $QIO
            00000004  0000       INDENT          = 4               ; Spaces to indent slave's log on copy
```

```
                                       0000          .SBTTL  Read-Only Data
                               00000000          .PSECT  RODATA,NOEXE,NOWRT,PAGE
                                       0000
                                       0000     PROCESS_NAME:                             ; Test and image name
49 4C 43 54 45 55 00000008'010E0000'   0000          .ASCID  /UETCLIG00/
                             30 30 47  000E
                                       0011
                                       0011     SYS$INPUT:                                ; Name of device from which...
4E 49 24 53 59 53 00000019'010E0000'   0011          .ASCID  /SYS$INPUT/              ; ...the test can be aborted
                             54 55 50  001F
                                       0022
                                       0022     SYS$NET:                                  ; Logical name of DECnet link...
45 4E 24 53 59 53 0000002A'010E0000'   0022          .ASCID  /SYS$NET/                ; ...if we're a network process
                                   54  0030
                                       0031
                                       0031     REPORT:                                   ; Tells us the type of regular...
54 52 4F 50 45 52 00000039'010E0000'   0031          .ASCID  /REPORT/                 ; ...messages to type to SYS$OUTPUT
                                       003F
                                       003F     SHORT:                                    ; If translation of REPORT, says...
   54 52 4F 48 53 00000047'010E0000'   003F          .ASCID  /SHORT/                  ; ...to type minimal non-error messages
                                       004C
                                       004C     MODE:                                     ; If defined as "DUMP" says to type...
      45 44 4F 4D 00000054'010E0000'   004C          .ASCID  /MODE/                   ; ...tracing messages as we progress
                                       0058
                                       0058     DUMP:                                     ; String to match for dump mode...
      50 4D 55 44 00000060'010E0000'   0058          .ASCID  /DUMP/                   ; ...operation
                                       0064
                                       0064     OPA0:                                     ; Name of device to receive warning...
   3A 30 41 50 4F 0000006C'010E0000'   0064          .ASCID  /OPA0:/                  ; ...of testing on other nodes
                                       0071
                                       0071     TASK:                                     ; Used to set up DECnet link...
45 54 53 59 53 22 00000079'010E0000'   0071          .ASCID  /"SYSTEST_CLIG"::"TASK=UETCLIG00"/ ; ...if we're master process
54 22 3A 3A 22 47 49 4C 43 5F 54 53    007F
30 47 49 4C 43 54 45 55 3D 4B 53 41    008B
                                22 30  0097
                                       0099
                                       0099     VMS:                                      ; SWTYPE in system block that we want
                          20 53 4D 56  0099          .ASCII  /VMS /
                                       009D
                                       009D     UETCLIG:                                  ; Becomes part of a slave's process name
49 4C 43 54 45 55 000000A5'010E0000'   009D          .ASCID  /UETCLIG_/
                                5F 47  00AB
                                       00AD
                                       00AD     MASTER:                                   ; Fills in READ_MSG, WRITE_MSG...
72 65 74 73 61 6D 000000B5'010E0000'   00AD          .ASCID  /master/                 ; ...GARBLE_MSG and NEWNAM
                                       00BB
                                       00BB     NULL:                                     ; Fills in READ_MSG, WRITE_MSG...
                             00000000  00BB          .LONG   0                        ; ...and GARBLE_MSG
                                       00BF
                                       00BF     BLANK_LINE:                               ; Puts white space on a page
                  000000C7'010E0000'   00BF          .ASCID  //
                                       00C7
                                       00C7     UETP$CLIG:                                ; Part of a test filespec...
43 24 50 54 45 55 000000CF'010E0000'   00C7          .ASCID  /UETP$CLIG_/             ; ...and part of lock names
                          5F 47 49 4C  00D5
                                       00D9
                                       00D9     BLOCK:                                    ; Part of a lock RESNAM when using...
```

```
4B 43 4F 4C 42 5F 000000E1'010E0000'  00D9    285             .ASCID  /_BLOCK/                        ; ...blocking ASTs
                                      00E7    286
                                      00E7    287     DOTTEST:                                        ; Part of a test filespec
3B 54 53 45 54 2E 000000EF'010E0000'  00E7    288             .ASCID  /.TEST;1/
                                  31  00F5
                                      00F6    289
                                      00F6    290     SYSTEST_DIR:                                    ; Part of a test filespec (default)
45 54 53 59 53 5B 000000FE'010E0000'  00F6    291             .ASCID  /[SYSTEST]/
                            5D 54 53  0104
                                      0107    292
                                      0107    293     SYS0_SYSTEST_DIR:                               ; Part of a test filespec (default)
2E 30 53 59 53 5B 0000010F'010E0000'  0107    294             .ASCID  /[SYS0.SYSTEST]/
         5D 54 53 45 54 53 59 53      0115
                                      011D    295
                                      011D    296     FILE:                                           ; Fills in RMS_ERR_STRING
      65 6C 69 66 00000125'010E0000'  011D    297             .ASCID  /file/
                                      0129    298
                                      0129    299     RECORD:                                         ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 00000131'010E0000'  0129    300             .ASCID  /record/
                                      0137    301
                                      0137    302     RMS_ERR_STRING:                                 ; Announces an RMS error
41 21 20 53 4D 52 0000013F'010E0000'  0137    303             .ASCID  /RMS !AS error in file !AD/
66 20 6E 69 20 72 6F 72 72 65 20 53   0145
            44 41 21 20 65 6C 69      0151
                                      0158    304
                                      0158    305     STATUS_STRING:                                  ; Announces text for a status value
73 75 74 61 74 53 00000160'010E0000'  0158    306             .ASCID  /Status returned was, "/
61 77 20 64 65 6E 72 75 74 65 72 20   0166
                         22 20 2C 73  0172
                                      0176    307
                                      0176    308     LONELY_MSG:                                     ; We're a solitary system
73 20 73 69 68 54 0000017E'010E0000'  0176    309             .ASCID  /This system is not a member of any cluster./
74 6F 6E 20 73 69 20 6D 65 74 73 79   0184
66 6F 20 72 65 62 6D 65 6D 20 61 20   0190
72 65 74 73 75 6C 63 20 79 6E 61 20   019C
                                  2E  01A8
                                      01A9    310
                                      01A9    311     REBEL_MSG:                                      ; Tells if CI occupant not in cluster
73 69 20 53 41 21 000001B1'010E0000'  01A9    312             .ASCID  /!AS is not a member of the cluster./
65 62 6D 65 6D 20 61 20 74 6F 6E 20   01B7
75 6C 63 20 65 68 74 20 66 6F 20 72   01C3
                      2E 72 65 74 73  01CF
                                      01D4    313
                                      01D4    314     WARN_OF_TESTING:                                ; Warns cluster OPA0s of our test
65 74 6F 4E 2F 21 000001DC'010E0000'  01D4    315             .ASCID  \!/Note to Operator:\-
72 6F 74 61 72 65 70 4F 20 6F 74 20   01E2
                                  3A  01EE
75 6C 43 20 50 54 45 55 5F 21 2F 21   01EF    316                     \!/!_UETP Cluster Integration Test started by node !AD at !%D.\
61 72 67 65 74 6E 49 20 72 65 74 73   01FB
74 73 20 74 73 65 54 20 6E 6F 69 74   0207
64 6F 6E 20 79 62 20 64 65 74 72 61   0213
44 25 21 20 74 61 20 44 41 21 20 65   021F
                                  2E  022B
                                      022C    317
                                      022C    318     END_OF_TESTING:                                 ; Tells cluster OPA0s of test end
65 74 6F 4E 2F 21 00000234'010E0000'  022C    319             .ASCID  \!/Note to Operator:\-
72 6F 74 61 72 65 70 4F 20 6F 74 20   023A
```

```
                                                               E 7
UETCLIG00                          VAX/VMS UETP Cluster Integration Test    16-SEP-1984 00:19:09  VAX/VMS Macro V04-00        Page   8
V04-000                            Read-Only Data                            6-SEP-1984 10:00:47  [UETPSY.SRC]UETCLIG00.MAR;1        (3)

                                 3A  0246
75 6C 43 20 50 54 45 55 5F 21 2F 21  0247    320                  \!/!_UETP Cluster Integration Test ended by node !AD at !%D.\
61 72 67 65 74 6E 49 20 72 65 74 73  0253
6E 65 20 74 73 65 54 20 6E 6F 69 74  025F
20 65 64 6F 6E 20 79 62 20 64 65 64  026B
   2E 44 25 21 20 74 61 20 44 41 21  0277
                                     0282    321
                                     0282    322 BRKTHRU_ERRORS:                           ; Warnings didn't get to all OPA0s
70 6F 20 57 55 21 0000028A'010E0000' 0282    323         .ASCID  \!UW operator console!%S timed out on the cluster test warning\-
6F 73 6E 6F 63 20 72 6F 74 61 72 65  0290
20 64 65 6D 69 74 20 53 25 21 65 6C  029C
63 20 65 68 74 20 6E 6F 20 74 75 6F  02A8
20 74 73 65 74 20 72 65 74 73 75 6C  02B4
                  67 6E 69 6E 72 61 77  02C0
20 57 55 21 20 64 6E 61 5F 21 2F 21  02C7    324                  \!/!_and !UW operator console!%S rejected it.\
6E 6F 63 20 72 6F 74 61 72 65 70 6F  02D3
65 6A 65 72 20 53 25 21 65 6C 6F 73  02DF
            2E 74 69 20 64 65 74 63  02EB
                                     02F3    325
                                     02F3    326 CLSIODB_FAIL:                             ; UETP$CLSIODB returned an error
65 6C 62 61 6E 55 000002FB'010E0000' 02F3    327         .ASCID  /Unable to read list of cluster nodes and devices./
73 69 6C 20 64 61 65 72 20 6F 74 20  0301
72 65 74 73 75 6C 63 20 66 6F 20 74  030D
64 20 64 6E 61 20 73 65 64 6F 6E 20  0319
            2E 73 65 63 69 76 65  0325
                                     032C    328
                                     032C    329 CLSIODB_SCREWEY:                          ; Record was not a system block record
6E 72 65 74 6E 49 00000334'010E0000' 032C    330         .ASCID  /Internal list of cluster nodes is inconsistent./
63 20 66 6F 20 74 73 69 6C 20 6C 61  033A
73 65 64 6F 6E 20 72 65 74 73 75 6C  0346
73 69 73 6E 6F 63 6E 69 20 73 69 20  0352
               2E 74 6E 65 74  035E
                                     0363    331
                                     0363    332 LINK_FAILED:                              ; $ASSIGN failed
20 64 6C 75 6F 43 0000036B'010E0000' 0363    333         .ASCID  \Could not set up a DECnet link to !AS.  Please check the\-
61 20 70 75 20 74 65 73 20 74 6F 6E  0371
6B 6E 69 6C 20 74 65 6E 43 45 44 20  037D
6C 50 20 20 2E 53 41 21 20 6F 74 20  0389
74 20 6B 63 65 68 63 20 65 73 61 65  0395
                                  65 68  03A1
63 6F 64 20 50 54 45 55 5F 21 2F 21  03A3    334                  \!/!_UETP documentation for the correct cluster preparation.\-
66 20 6E 6F 69 74 61 74 6E 65 6D 75  03AF
65 72 72 6F 63 20 65 68 74 20 72 6F  03BB
70 20 72 65 74 73 75 6C 63 20 74 63  03C7
   2E 6E 6F 69 74 61 72 61 70 65 72  03D3
53 41 21 20 65 64 6F 4E 5F 21 2F 21  03DE    335                  \!/!_Node !AS will not be included in cluster lock testing.\
65 62 20 74 6F 6E 20 6C 6C 69 77 20  03EA
6E 69 20 64 65 64 75 6C 63 6E 69 20  03F6
63 6F 6C 20 72 65 74 73 75 6C 63 20  0402
      2E 67 6E 69 74 73 65 74 20 6B  040E
                                     0418    336
                                     0418    337 NO_NODE_MSG:                              ; No nodes found to be testable
61 76 61 20 6F 4E 00000420'010E0000' 0418    338         .ASCID  \No available cluster DECnet/VAX nodes found for lock tests.\
74 73 75 6C 63 20 65 6C 62 61 6C 69  0426
41 56 2F 74 65 6E 43 45 44 20 72 65  0432
6E 75 6F 66 20 73 65 64 6F 6E 20 58  043E
74 20 6B 63 6F 6C 20 72 6F 66 20 64  044A
```

```
                  2E 73 74 73 65  0456
                                  045B   339
                                  045B   340 NODE_LIST_MSG:                        ; Names nodes to test
20 73 65 64 6F 4E 00000463'010E0000'  045B   341         .ASCID  /Nodes included in lock tests:  !#(AS)/
20 6E 69 20 64 65 64 75 6C 63 6E 69  0469
20 3A 73 74 73 65 74 20 6B 63 6F 6C  0475
            29 53 41 28 23 21 20  0481
                                  0488   342
                                  0488   343 COMMASPACE:                           ; Separates successive nodes...
            20 2C 00000490'010E0000'  0488   344         .ASCID  /, /                  ; ...for NODE_LIST_MSG
                                  0492   345
                                  0492   346 CRLFTAB:                              ; Wraps a line for NODE_LIST_MSG
         09 0A 0D 0000049A'010E0000'  0492   347         .ASCID  <13><10>/       /
                                  049D   348
                                  049D   349 WRONG_ENQ:                            ; $ENQ for master's lock goofed
6F 20 51 4E 45 24 000004A5'010E0000'  049D   350         .ASCID  \$ENQ of a lock that should have been owned by a process\-
61 68 74 20 6B 63 6F 6C 20 61 20 66  04AB
76 61 68 20 64 6C 75 6F 68 73 20 74  04B7
64 65 6E 77 6F 20 6E 65 65 62 20 65  04C3
73 65 63 6F 72 70 20 61 20 79 62 20  04CF
                                  73  04DB
20 67 6E 69 6E 6E 75 72 5F 21 2F 21  04DC   351                 \!/!_running on !AS got an unexpected result (below).\-
61 20 74 6F 67 20 53 41 21 20 6E 6F  04E8
64 65 74 63 65 70 78 65 6E 75 20 6E  04F4
6C 65 62 28 20 74 6C 75 73 65 72 20  0500
                     2E 29 77 6F  050C
75 73 65 72 20 65 68 54 5F 21 2F 21  0510   352                 \!/!_The result should have been "SYSTEM-W-NOTQUEUED".\
61 68 20 64 6C 75 6F 68 73 20 74 6C  051C
53 59 53 22 20 6E 65 65 62 20 65 76  0528
45 55 51 54 4F 4E 2D 57 2D 4D 45 54  0534
                  2E 22 44 45 55  0540
                                  0545   353
                                  0545   354 NO_LOCK_ENQ:                          ; Slave couldn't get a lock it wanted
6F 20 51 4E 45 24 0000054D'010E0000'  0545   355         .ASCID  \$ENQ of a lock that should have been available failed.\
61 68 74 20 6B 63 6F 6C 20 61 20 66  0553
76 61 68 20 64 6C 75 6F 68 73 20 74  055F
6C 69 61 76 61 20 6E 65 65 62 20 65  056B
2E 64 65 6C 69 61 66 20 65 6C 62 61  0577
                                  0583   356
                                  0583   357 NO_BLOCK_LOCK:                        ; Master can't do $ENQ with BLKAST set
65 6C 62 61 6E 55 0000058B'010E0000'  0583   358         .ASCID  \Unable to set up a lock to check blocking ASTs in deadlock \-
61 20 70 75 20 74 65 73 20 6F 74 20  0591
65 68 63 20 6F 74 20 6B 63 6F 6C 20  059D
20 67 6E 69 6B 63 6F 6C 62 20 6B 63  05A9
64 61 65 64 20 6E 69 20 73 54 53 41  05B5
                  20 6B 63 6F 6C  05C1
                  2E 74 73 65 74  05C6   359                 \test.\
                                  05CB   360 NO_DLOCK_SETUP:                       ; Node died during deadlock setup
20 70 75 74 65 53 000005D3'010E0000'  05CB   361         .ASCID  \Setup for deadlock testing may have been broken.\-
6B 63 6F 6C 64 61 65 64 20 72 6F 66  05D9
79 61 6D 20 67 6E 69 74 73 65 74 20  05E5
62 20 6E 65 65 62 20 65 76 61 68 20  05F1
                  2E 6E 65 6B 6F 72  05FD
69 64 20 65 73 61 65 6C 50 09 0A 0D  0603   362                 <13><10>\       Please disregard any deadlock error message.\
20 79 6E 61 20 64 72 61 67 65 72 73  060F
72 72 65 20 6B 63 6F 6C 64 61 65 64  061B
   2E 65 67 61 73 73 65 6D 20 72 6F  0627
```

```
                                                                   G  7
UETCLIG00                              VAX/VMS UETP Cluster Integration Test    16-SEP-1984 00:19:09  VAX/VMS Macro V04-00         Page  10
V04-000                                Read-Only Data                            6-SEP-1984 10:00:47  [UETPSY.SRC]UETCLIG00.MAR;1        (3)

                                                     0632   363
                                                     0632   364 DEADLOCK_OFF_MSG:                        ; Someone has d'lock detection disabled
6F 6C 64 61 65 44 0000063A'010E0000'  0632   365         .ASCID  \Deadlock detection is disabled on !AD.\
6E 6F 69 74 63 65 74 65 64 20 6B 63  0640
64 65 6C 62 61 73 69 64 20 73 69 20  064C
            2E 44 41 21 20 6E 6F 20  0658
                                                     0660   366
                                                     0660   367 DEADLOCK_WAIT_MSG:                       ; DEADLOCK_WAIT was inconsistent
6F 6C 64 61 65 44 00000668'010E0000'  0660   368         .ASCID  \Deadlock checking interval is !UL second!%S on !AS,\-
20 67 6E 69 6B 63 65 68 63 20 6B 63  066E
20 73 69 20 6C 61 76 72 65 74 6E 69  067A
25 21 64 6E 6F 63 65 73 20 4C 55 21  0686
         2C 53 41 21 20 6E 6F 20 53  0692
20 4C 55 21 20 74 75 62 5F 21 2F 21  069B   369                 \!/!_but !UL second!%S on !AD.\
6E 6F 20 53 25 21 64 6E 6F 63 65 73  06A7
               2E 44 41 21 20  06B3
                                                     06B8   370
                                                     06B8   371 VICTIMS_MSG:                             ; Problem with deadlock detection
69 76 20 4C 55 21 000006C0'010E0000'  06B8   372         .ASCID  \!UL victim!%S chosen for cluster-wide deadlock detection.\
73 6F 68 63 20 53 25 21 6D 69 74 63  06C6
74 73 75 6C 63 20 72 6F 66 20 6E 65  06D2
64 61 65 64 20 65 64 69 77 2D 72 65  06DE
69 74 63 65 74 65 64 20 6B 63 6F 6C  06EA
                        2E 6E 6F  06F6
                                                     06F9   373
                                                     06F9   374 DLOCK_ENQ:                               ; Slave couldn't queue a lock request
66 20 51 4E 45 24 00000701'010E0000'  06F9   375         .ASCID  \$ENQ failed to queue a request during deadlock test.\
65 75 71 20 6F 74 20 64 65 6C 69 61  0707
74 73 65 75 71 65 72 20 61 20 65 75  0713
64 61 65 64 20 67 6E 69 72 75 64 20  071F
      2E 74 73 65 74 20 6B 63 6F 6C  072B
                                                     0735   376
                                                     0735   377 NO_SLAVE_BLOCK:                          ; Slave's blocked lock request failed
67 20 51 4E 45 24 0000073D'010E0000'  0735   378         .ASCID  \$ENQ got unexpected result for resource for which BLKAST was \-
65 74 63 65 70 78 65 6E 75 20 74 6F  0743
72 6F 66 20 74 6C 75 73 65 72 20 64  074F
6F 66 20 65 63 72 75 6F 73 65 72 20  075B
41 4B 4C 42 20 68 63 69 68 77 20 72  0767
            20 73 61 77 20 54 53  0773
         2E 64 65 6C 62 61 6E 65  077A   379                 \enabled.\
                                                     0782   380
                                                     0782   381 MEMB_PATH:                               ; Can't attempt file access
74 61 20 74 6F 4E 0000078A'010E0000'  0782   382         .ASCID  \Not attempting file test to !AD.\-
6C 69 66 20 67 6E 69 74 70 6D 65 74  0790
41 21 20 6F 74 20 74 73 65 74 20 65  079C
                              2E 44  07A8
20 73 69 20 65 64 6F 4E 5F 21 2F 21  07AA   383                 \!/!_Node is not a cluster member or path to it is not enabled.\
65 74 73 75 6C 63 20 61 20 74 6F 6E  07B6
20 72 6F 20 72 65 62 6D 65 6D 20 72  07C2
69 20 74 69 20 6F 74 20 68 74 61 70  07CE
65 6C 62 61 6E 65 20 74 6F 6E 20 73  07DA
                              2E 64  07E6
                                                     07E8   384
                                                     07E8   385 NO_FILE_NODE:                            ; All $CREATEs failed
69 75 73 20 6F 4E 000007F0'010E0000'  07E8   386         .ASCID  /No suitable disk found to check remote file access on !AD./
66 20 6B 73 69 64 20 65 6C 62 61 74  07F6
63 65 68 63 20 6F 74 20 64 6E 75 6F  0802
```

```
6C 69 66 20 65 74 6F 6D 65 72 20 6B  080E
20 6E 6F 20 73 73 65 63 63 61 20 65  081A
                        2E 44 41 21  0826
                                     082A
                                     082A      SLAVE_NO_ACCESS:                      ; Can't get to shared file
73 65 63 6F 72 50 00000832'010E0000' 082A              .ASCID  \Process on !AS was unable to share access to !AS.\
73 61 77 20 53 41 21 20 6E 6F 20 73  0838
73 20 6F 74 20 65 6C 62 61 6E 75 20  0844
20 73 73 65 63 63 61 20 65 72 61 68  0850
               2E 53 41 21 20 6F 74  085C
                                     0863
                                     0863      SLAVE_EXT_FAIL:                       ; Error reading second block
73 65 63 6F 72 50 0000086B'010E0000' 0863              .ASCID  \Process on !AS had trouble reading !AS when file was extended.\
64 61 68 20 53 41 21 20 6E 6F 20 73  0871
61 65 72 20 65 6C 62 75 6F 72 74 20  087D
65 68 77 20 53 41 21 20 67 6E 69 64  0889
65 20 73 61 77 20 65 6C 69 66 20 6E  0895
            2E 64 65 64 6E 65 74 78  08A1
                                     08A9
                                     08A9      WRITE_MSG:                            ; DECnet write $QIO failed
74 65 6E 43 45 44 000008B1'010E0000' 08A9              .ASCID  /DECnet write of "!AD" message to !AS failed.!AS/
21 22 20 66 6F 20 65 74 69 72 77 20  08B7
20 65 67 61 73 73 65 6D 20 22 44 41  08C3
65 6C 69 61 66 20 53 41 21 20 6F 74  08CF
                     53 41 21 2E 64  08DB
                                     08E0
                                     08E0      READ_MSG:                             ; DECnet read $QIO failed
74 65 6E 43 45 44 000008E8'010E0000' 08E0              .ASCID  /DECnet read of "!AD" message from !AS failed.!AS/
41 21 22 20 66 6F 20 64 61 65 72 20  08EE
66 20 65 67 61 73 73 65 6D 20 22 44  08FA
6C 69 61 66 20 53 41 21 20 6D 6F 72  0906
                  53 41 21 2E 64 65  0912
                                     0918
                                     0918      GARBLE_MSG:                           ; Node replied with trash to our message
65 6C 62 72 61 47 00000920'010E0000' 0918              .ASCID  /Garbled "!AD" message or unexpected message from !AS.!AS/
73 73 65 6D 20 22 44 41 21 22 20 64  0926
70 78 65 6E 75 20 72 6F 20 65 67 61  0932
67 61 73 73 65 6D 20 64 65 74 63 65  093E
21 2E 53 41 21 20 6D 6F 72 66 20 65  094A
                              53 41  0956
                                     0958
                                     0958      CANCEL_MSG:                           ; $QIO was $CANCELled on timed out chan
20 64 65 6D 69 54 00000960'010E0000' 0958              .ASCID  \Timed out on DECnet $QIO to/from !AS.  I/O was cancelled.\
65 6E 43 45 44 20 6E 6F 20 74 75 6F  0966
72 66 2F 6F 74 20 4F 49 51 24 20 74  0972
4F 2F 49 20 20 2E 53 41 21 20 6D 6F  097E
6C 6C 65 63 6E 61 63 20 73 61 77 20  098A
                           2E 64 65  0996
                                     0999
                                     0999      EXCLUDE_MSG:                          ; Consequence of DECnet error
61 68 54 09 0A 0D 000009A1'010E0000' 0999              .ASCID  <13><10>/       That node is excluded from further tests./
78 65 20 73 69 20 65 64 6F 6E 20 74  09A7
20 6D 6F 72 66 20 64 65 64 75 6C 63  09B3
74 73 65 74 20 72 65 68 74 72 75 66  09BF
                              2E 73  09CB
                                     09CD
                                     09CD      PLEASE_CHECK_MSG:                     ; Failure while copying slave's log
```

```
65 6C 50 09 0A 0D 000009D5'010E0000'  09CD             .ASCID  <13><10><9>\Please check SYS$TEST:NETSERVER.LOG on that node.\
59 53 20 6B 63 65 68 63 20 65 73 61   09DB
45 53 54 45 4E 3A 54 53 45 54 24 53   09E7
20 6E 6F 20 47 4F 4C 2E 52 45 56 52   09F3
      2E 65 64 6F 6E 20 74 61 68 74   09FF
                                      0A09
                                      0A09    DEBUG_INTRO_MSG:                          ; Warns that we'll report debugging info
20 65 63 61 72 74 00000A11'010E0000'  0A09             .ASCID  \trace -- Program execution trace messages are enabled.\
65 20 6D 61 72 67 6F 72 50 20 2D 2D   0A17
61 72 74 20 6E 6F 69 74 75 63 65 78   0A23
20 73 65 67 61 73 73 65 6D 20 65 63   0A2F
2E 64 65 6C 62 61 6E 65 20 65 72 61   0A3B
                                      0A47
                                      0A47    DEBUG_WRITE_MSG:                          ; Reports debugging info
20 65 63 61 72 74 00000A4F'010E0000'  0A47             .ASCID  \trace -- $QIO write of !AD message to !AS.\
74 69 72 77 20 4F 49 51 24 20 2D 2D   0A55
73 65 6D 20 44 41 21 20 66 6F 20 65   0A61
2E 53 41 21 20 6F 74 20 65 67 61 73   0A6D
                                      0A79
                                      0A79    DEBUG_READ_MSG:                           ; Reports debugging info
20 65 63 61 72 74 00000A81'010E0000'  0A79             .ASCID  \trace -- $QIO read of !AD message from !AS.\
64 61 65 72 20 4F 49 51 24 20 2D 2D   0A87
73 73 65 6D 20 44 41 21 20 66 6F 20   0A93
53 41 21 20 6D 6F 72 66 20 65 67 61   0A9F
                                  2E  0AAB
                                      0AAC
                                      0AAC    DEBUG_REQ_LOCK_MSG:                       ; Master told slave to take out lock
20 65 63 61 72 74 00000AB4'010E0000'  0AAC             .ASCID  \trace -- !AS was requested to lock resource !AS.\
72 20 73 61 77 20 53 41 21 20 2D 2D   0ABA
20 6F 74 20 64 65 74 73 65 75 71 65   0AC6
63 72 75 6F 73 65 72 20 6B 63 6F 6C   0AD2
                  2E 53 41 21 20 65   0ADE
                                      0AE4
                                      0AE4    DEBUG_TAK_LOCK_MSG:                       ; Slave is requesting a lock
20 65 63 61 72 74 00000AEC'010E0000'  0AE4             .ASCID  \trace -- Queuing up a lock for resource !AS.\
75 20 67 6E 69 75 65 75 51 20 2D 2D   0AF2
72 6F 66 20 6B 63 6F 6C 20 61 20 70   0AFE
41 21 20 65 63 72 75 6F 73 65 72 20   0B0A
                               2E 53  0B16
                                      0B18
                                      0B18    DEBUG_DLOCK_VICTIM_MSG:                   ; Slave was/was not selected as victim
20 65 63 61 72 74 00000B20'010E0000'  0B18             .ASCID  \trace -- !AD was !ASselected as the deadlock victim.\
21 20 73 61 77 20 44 41 21 20 2D 2D   0B26
61 20 64 65 74 63 65 6C 65 73 53 41   0B32
6F 6C 64 61 65 64 20 65 68 74 20 73   0B3E
      2E 6D 69 74 63 69 76 20 6B 63   0B4A
                                      0B54
                                      0B54    NOT_MSG:                                  ; Used to fill in DEBUG_DLOCK_VICTIM_MSG
      20 74 6F 6E 00000B5C'010E0000'  0B54             .ASCID  \not \
                                      0B60
                                      0B60    DEBUG_FILE_MSG:                           ; Reports debugging info
20 65 63 61 72 74 00000B68'010E0000'  0B60             .ASCID  \trace -- Created !AS.\
21 20 64 65 74 61 65 72 43 20 2D 2D   0B6E
                            2E 53 41  0B7A
                                      0B7D
                                      0B7D    DEBUG_NOFILE_MSG:                         ; Reports debugging info
20 65 63 61 72 74 00000B85'010E0000'  0B7D             .ASCID  \trace -- Failed to create !AS.  Status was !XL.\
6F 74 20 64 65 6C 69 61 46 20 2D 2D  0B8B
2E 53 41 21 20 65 74 61 65 72 63 20  0B97
73 61 77 20 73 75 74 61 74 53 20 20  0BA3
               2E 4C 58 21 20        0BAF
                                     0BB4
                                     0BB4   DEBUG_NOSHARE_MSG:                          ; Reports debugging info
20 65 63 61 72 74 00000BBC'010E0000' 0BB4           .ASCID  \trace -- No available node to share access to !AS.\
61 6C 69 61 76 61 20 6F 4E 20 2D 2D  0BC2
20 6F 74 20 65 64 6F 6E 20 65 6C 62  0BCE
73 73 65 63 63 61 20 65 72 61 68 73  0BDA
         2E 53 41 21 20 6F 74 20     0BE6
                                     0BEE
                                     0BEE   DEBUG_SHARE_MSG:                            ; Reports debugging info
20 65 63 61 72 74 00000BF6'010E0000' 0BEE           .ASCID  \trace -- !AD was able to share access to !AS.\
61 20 73 61 77 20 44 41 21 20 2D 2D  0BFC
65 72 61 68 73 20 6F 74 20 65 6C 62  0C08
21 20 6F 74 20 73 73 65 63 63 61 20  0C14
                         2E 53 41    0C20
                                     0C23
                                     0C23   DEBUG_EXTEND_MSG:                           ; Reports debugging info
20 65 63 61 72 74 00000C2B'010E0000' 0C23           .ASCID  \trace -- !AD read additional records when !AS was extended.\
20 64 61 65 72 20 44 41 21 20 2D 2D  0C31
72 20 6C 61 6E 6F 69 74 69 64 64 61  0C3D
20 6E 65 68 77 20 73 64 72 6F 63 65  0C49
65 74 78 65 20 73 61 77 20 53 41 21  0C55
               2E 64 65 64 6E        0C61
                                     0C66
                                     0C66   ABORTC_MSG_PTR:                             ; $PUTMSG MSGVEC for CTRL/C handler
                           000F 0003 0C66           .WORD   3,^XF
                            0074832B 0C6A           .LONG   UETP$_ABORTC!STS$K_SUCCESS
                           0000 0001 0C6E           .WORD   1,0
                           00000000' 0C72           .ADDRESS PROCESS_NAME
                                     0C76
                                     0C76   LONELY_MSG_PTR:                             ; $PUTMSG MSGVEC for not in a cluster
                           000F 0003 0C76           .WORD   3,^XF
                            00741133 0C7A           .LONG   UETP$_TEXT!STS$K_INFO
                           0000 0001 0C7E           .WORD   1,0
                           00000176' 0C82           .ADDRESS LONELY_MSG
                                     0C86
                                     0C86   REBEL_MSG_PTR:                              ; $PUTMSG MSGVEC for node not in cluster
                           000F 0003 0C86           .WORD   3,^XF
                            00741133 0C8A           .LONG   UETP$_TEXT!STS$K_INFO
                           0000 0001 0C8E           .WORD   1,0
                           00000CBC' 0C92           .ADDRESS BUFFER_PTR
                                     0C96
                                     0C96   NO_NODE_MSG_PTR:                            ; $PUTMSG MSGVEC for no nodes to test
                           000F 0003 0C96           .WORD   3,^XF
                            00741130 0C9A           .LONG   UETP$_TEXT!STS$K_WARNING
                           0000 0001 0C9E           .WORD   1,0
                           00000418' 0CA2           .ADDRESS NO_NODE_MSG
                                     0CA6
                                     0CA6   NODE_LIST_MSG_PTR:                          ; $PUTMSG MSGVEC for nodes to test
                           0001 0003 0CA6           .WORD   3,^X1
                            00741133 0CAA           .LONG   UETP$_TEXT!STS$K_INFO
                           0000 0001 0CAE           .WORD   1,0
                           00000CBC' 0CB2           .ADDRESS BUFFER_PTR
                                     0CB6
```

```
                   0CB6   NO_DLOCK_SETUP_PTR:                              ; $PUTMSG MSGVEC for deadlock...
        000F 0003  0CB6           .WORD   3,^XF                            ; ...setup problems
         00741130  0CBA           .LONG   UETP$_TEXT!STS$K_WARNING
        0000 0001  0CBE           .WORD   1,0
        000005CB'  0CC2           .ADDRESS NO_DLOCK_SETUP
                   0CC6
                   0CC6   DEADLOCK_OFF_PTR:                                ; $PUTMSG MSGVEC if some node has...
                   0CC6                                                    ; deadlock detection disabled
                   0CC6   MEMB_PATH_PTR:                                   ; $PUTMSG MSGVEC for case when can't...
                   0CC6                                                    ; ...do file access on a node because...
                   0CC6                                                    ; ...the node is not a cluster member...
                   0CC6                                                    ; ...or has no useable path to it
                   0CC6   NO_FILE_NODE_PTR:                                ; $PUTMSG MSGVEC for case when can't...
                   0CC6                                                    ; ...create test file on some node
                   0CC6   CANCEL_MSG_PTR:                                  ; $PUTMSG MSGVEC for $CANCEL $QIO
        000F 0003  0CC6           .WORD   3,^XF
         00741130  0CCA           .LONG   UETP$_TEXT!STS$K_WARNING
        0000 0001  0CCE           .WORD   1,0
        00000CBC'  0CD2           .ADDRESS BUFFER_PTR
                   0CD6
                   0CD6   BLANK_LINE_PTR:                                  ; $PUTMSG MSGVEC for leaving...
        0001 0003  0CD6           .WORD   3,^X1                            ; ...a blank line between messages
         00741131  0CDA           .LONG   UETP$_TEXT!STS$K_SUCCESS         ; Note that if we incorporate this...
        0000 0001  0CDE           .WORD   1,0                              ; ...into another MSGVEC, the "%"...
        000000BF'  0CE2           .ADDRESS BLANK_LINE                      ; ...of that message becomes a "-"
                   0CE6
                   0CE6   ERRORLOG_PTR:                                    ; $PUTMSG MSGVEC for copying...
        0001 0004  0CE6           .WORD   4,^X1                            ; ... a slave's SYS$ERROR.LOG
         007480B9  0CEA           .LONG   UETP$_COPY_LOG_LINE
        0000 0002  0CEE           .WORD   2,0
         00000000  0CF2           .LONG   0
        00000CBC'  0CF6           .ADDRESS BUFFER_PTR
                   0CFA
                   0CFA   DEBUG_QIO_MSG_PTR:                               ; $PUTMSG MSGVEC for $QIO debug msg
        000F 0003  0CFA           .WORD   3,^XF
         00741133  0CFE           .LONG   UETP$_TEXT!STS$K_INFO
        0000 0001  0D02           .WORD   1,0
        00000FF3'  0D06           .ADDRESS DEBUG_PTR
                   0D0A
                   0D0A   INPUT_ITMLST:                                    ; $GETDVI arg list for SYS$INPUT
        0020 0040  0D0A           .WORD   64,DVI$_DEVNAM                   ; We need the equivalence name...
00000CBC'00000CC4'  0D0E           .ADDRESS BUFFER,BUFFER_PTR
        0002 0004  0D16           .WORD   4,DVI$_DEVCHAR                   ; ...and the device independent info
00000000'0000003E'  0D1A           .ADDRESS DEVCHAR,0
         00000000  0D22           .LONG   0
                   0D26
                   0D26   MYNODE_ITMLST:                                   ; $GETSYI arg list for...
        1067 0006  0D26           .WORD   NODE_LENGTH,SYI$_SCSNODE         ; ...my node name...
00000000'00000042'  0D2A           .ADDRESS SCSNODE,0
        105E 0004  0D32           .WORD   4,SYI$_DEADLOCK_WAIT             ; ...deadlock search interval
00000000'0000007C'  0D36           .ADDRESS DEADLOCK_WAIT,0
         00000000  0D3E           .LONG   0
                   0D42
                   0D42   OTHERNODE_ITMLST:                                ; $GETSYI arg list for...
        10CF 0004  0D42           .WORD   4,SYI$_CLUSTER_MEMBER            ; ...cluster membership
00000000'00000090'  0D46           .ADDRESS CLUSTER_MEMBER,0
         00000000  0D4E           .LONG   0
```

```
                                    0D52
                                    0D52   MYPROC_ITMLST:                                  ; $GETJPI arg list for...
                         031C 000F  0D52           .WORD   PRCNAM_LENGTH,JPI$_PRCNAM ; ...my process name
               0000004A'00000052'   0D56           .ADDRESS CURNAM,CURNAM_DESC
                          00000000  0D5E           .LONG   0
                                    0D62
                                    0D62   CLSIODB_ARGS:                                   ; Arg list when calling UETP$CLSIODB
                          00000004  0D62           .LONG   4
      00000000'00000000'000000A2'   0D66           .ADDRESS CLSPTR,0,0
                          0000002F  0D72           .LONG   UIDFLAG$M_SID!UIDFLAG$M_PATH!-
                                    0D76                   UIDFLAG$M_DDB!UIDFLAG$M_UCB!UIDFLAG$M_MYSYS
                                    0D76
                                    0D76   QIO_DELTA:                                      ; Delta time to wait for ordinary...
                 FFFFFFFF DC3CBA00  0D76           .LONG   -10000000*QIO_TIMEOUT,-1 ; ...DECnet $QIO completion
                                    0D7E
                                    0D7E   SLAVE_QIO_DELTA:                                ; Delta time to wait for slave...
                 FFFFFFFF 4D2FA200  0D7E           .LONG   -10000000*5*QIO_TIMEOUT,-1 ; ...read DECnet $QIO completion
                                    0D86                                                   ; They must be more tolerant...
                                    0D86                                                   ; ...because master services several
                                    0D86
                                    0D86   FIVE_SECONDS:                                   ; Nominal time to wait for $QIO when...
                 FFFFFFFF FD050F80  0D86           .LONG   -50000000,-1                    ; ...copying slave's error log to master
                                    0D8E
                                    0D8E   FAO_BUF:                                        ; Fixed desc for misc text strings
                          0000010D  0D8E           .LONG   TEXTB_SIZE
                          00000CC4' 0D92           .ADDRESS BUFFER
                                    0D96
                                    0D96   DEBUG_FAO_BUF:                                  ; Fixed desc for debug text strings
                          0000010D  0D96           .LONG   TEXTB_SIZE
                          00000FFB' 0D9A           .ADDRESS DEBUG_BUFFER
                                    0D9E
                                    0D9E   NO_RMS_AST_TABLE:                               ; List of errors for which...
                          00000000' 0D9E           .LONG   RMS$_BLN                        ; ...RMS cannot deliver an AST...
                          00000000' 0DA2           .LONG   RMS$_BUSY                       ; ...even if one has an ERR= arg
                          00000000' 0DA6           .LONG   RMS$_CDA                        ; Note that we can search table...
                          00000000' 0DAA           .LONG   RMS$_FAB                        ; ...via MATCHC since <31:16>...
                          00000000' 0DAE           .LONG   RMS$_RAB                        ; ...pattern can't be in <15:0>
                          00000014  0DB2   NRAT_LENGTH = .-NO_RMS_AST_TABLE
                                    0DB2
                                    0DB2   MESSAGE_NAMES:                                  ; Create message names and texts
                                    0DB2           .MACRO  DEFMSG  MSGNAM                  ; Define the way we'll name messages
                                    0DB2           MSGNAM'_MSG:
                                    0DB2                   .WORD   MSGNAM'_LENGTH
                                    0DB2                   .ASCII  /MSGNAM/
                                    0DB2           .ENDM   DEFMSG
                                    0DB2           MESSAGES                                ; Name and list messages with text
```

```
                  0E1C         .SBTTL  Read/Write Data
              00000000         .PSECT  RWDATA,WRT,NOEXE,PAGE
                  0000
                  0000 CLIG_ANNOUNCE:                               ; $PUTMSG MSGVEC for begin & end msgs
     000F 0004    0000         .WORD   4,^XF
      0074103B    0004         .LONG   UETP$_BEGIND!STS$K_INFO  ; This will change at test end
     0000 0002    0008         .WORD   2,0
     00000000'    000C         .ADDRESS PROCESS_NAME            ; This will change to new process name
     00000000     0010         .LONG   0
                  0014
                  0014 EXIT_DESC:                                   ; Exit handler descriptor
     00000000     0014         .LONG   0
     00001E8D'    0018         .ADDRESS EXIT_HANDLER
     00000001     001C         .LONG   1
     00000028'    0020         .ADDRESS EXIT_STATUS
                  0024
                  0024 FLAGS:                                       ; State variables existing over time
     00000028     0024         .BLKL   1                        ; (See Equated Symbols for definitions)
                  0028
                  0028 EXIT_STATUS:                                 ; Status value on program exit
     0000002C     0028         .BLKL   1
                  002C
                  002C QUAD_STATUS:                                 ; IO status block for misc sys. svcs.
     00000034     002C         .BLKQ   1
                  0034
                  0034 ERROR_COUNT:                                 ; Cumulative error count
     00000038     0034         .BLKL   1
                  0038
                  0038 ARG_COUNT:                                   ; Argument counter used by ERROR_EXIT
     0000003C     0038         .BLKL   1
                  003C
                  003C TTCHAN:                                      ; Channel associated with ctrl. term.
     0000003E     003C         .BLKW   1
                  003E
                  003E DEVCHAR:                                     ; Device independent characteristics
     00000042     003E         .BLKL   1
                  0042
                  0042 SCSNODE:                                     ; My node name in the cluster...
     0000004A     0042         .BLKL   2
                  004A
                  004A CURNAM_DESC:                                 ; Gets my process name length...
     0000004E     004A         .BLKW   2
     00000052'    004E         .ADDRESS CURNAM                  ; ...to become a descriptor
                  0052
                  0052 CURNAM:                                      ; My process name on entry
     00000061     0052         .BLKB   PRCNAM_LENGTH
                  0061
                  0061 NEWNAM_DESC:                                 ; Desc for the process name...
     00000065     0061         .BLKW   2
     00000069'    0065         .ADDRESS NEWNAM                  ; ...in use while running this image
                  0069
                  0069 NEWNAM:                                      ; My process name while running
     00000078     0069         .BLKB   PRCNAM_LENGTH
                  0078
                  0078 DEADLOCK_VICTIMS:                            ; Number of deadlock victim processes
     0000007C     0078         .BLKL   1
                  007C
```

```
                      007C  DEADLOCK_WAIT:                                          ; Deadlock search interval in seconds
          00000080    007C          .BLKL   1
                      0080
                      0080  DEADLOCK_COUNT:                                         ; Count of processes participating in...
          00000084    0080          .BLKL   1                                       ; ...a deadlock, but who have not yet...
                      0084                                                          ; ...caused a blocking AST for our...
                      0084                                                          ; ...lock used for communication
                      0084
                      0084  DEADLOCK_LOCKID:                                        ; Lock id of the lock used for...
          00000088    0084          .BLKL   1                                       ; ...blocking AST communication
                      0088
                      0088  DEADLOCK_MSG_TIME:                                      ; Delta time to wait to hear that...
          00000090    0088          .BLKQ   1                                       ; ...some process is a deadlock victim
                      0090
                      0090  CLUSTER_MEMBER:                                         ; Receives TRUE/FALSE if a VMS node...
          00000094    0090          .BLKL   1                                       ; ...is a member of our cluster
                      0094
                      0094  MASTER_NODE_DESC:                                       ; Simplifies using MASTER_NODE...
          00000006    0094          .LONG   NODE_LENGTH                             ; ...in $FAO strings
          0000009C'   0098          .ADDRESS MASTER_NODE
                      009C  MASTER_NODE:                                            ; Name of master node. This gets...
72 65 74 73 61 6D     009C          .ASCII  /master/                                ; ...overwritten when HELLO msg read
                      00A2
                      00A2  CLSPTR:                                                 ; Pointer to local copy of cluster db
          000000AA    00A2          .BLKL   2
                      00AA
                      00AA  NODE_CHANS:                                             ; List of DECnet channels to...
          000002A8    00AA          .BLKW   MAX_NODES                               ; ...nodes on which we have slaves
          000002AA    02A8          .BLKW   1                                       ; Guaranteed list terminator
                      02AA
                      02AA  NODE_NAMES:                                             ; List of descriptors to names of...
          00000AA2    02AA          .BLKQ   MAX_NODES                               ; ...nodes on which we have slaves
                      0AA2                                                          ; The second word of each descriptor...
                      0AA2                                                          ; ...carries flags.  No flags set...
                      0AA2                                                          ; ...(valid string descriptor) is the...
                      0AA2                                                          ; ...normal state
                      0AA2
                      0AA2  MESSAGE_BUFFER:                                         ; Messages we send to slave nodes...
          00000CBC    0AA2          .BLKB   2*TEXTB_SIZE                            ; ...or messages we receive from master
                      0CBC                                                          ; The size is to allow us to use...
                      0CBC                                                          ; ...this buffer to send a slave's...
                      0CBC                                                          ; ...copy of SYS$ERROR to the master
                      0CBC
                      0CBC  BUFFER_PTR:                                             ; Variable desc for misc text strings
          00000CC0    0CBC          .BLKL   1
          00000CC4'   0CC0          .ADDRESS BUFFER
                      0CC4  BUFFER:                                                 ; Buffer for miscellaneous text strings
          00000EDE    0CC4          .BLKB   2*TEXTB_SIZE                            ; The size is to allow us to use...
                      0EDE                                                          ; ...this buffer to send a slave's...
                      0EDE                                                          ; ...copy of SYS$ERROR to the master
                      0EDE
                      0EDE  STATUS_PTR:                                             ; Variable desc for status code strings
          00000EE2    0EDE          .BLKL   1
          00000EE6'   0EE2          .ADDRESS STATUS_BUFFER
                      0EE6  STATUS_BUFFER:
          00000FF3    0EE6          .BLKB   TEXTB_SIZE
                      0FF3
```

```
                   0FF3   696 DEBUG_PTR:                                          ; Variable desc for debug text strings
        00000FF7   0FF3   697           .BLKL   1
        00000FFB'  0FF7   698           .ADDRESS DEBUG_BUFFER
                   0FFB   699 DEBUG_BUFFER:
        0000142F   0FFB   700           .BLKL   TEXTB_SIZE
```

```
            142F         .SBTTL  RMS-32 Data Structures
            142F         .ALIGN  LONG
            1430
            1430  SE_FAB:                                          ; Used for copy of slave's SYS$ERROR
            1430         $FAB-
            1430         FNM = <SYS$ERROR.LOG>,-
            1430         NAM = SE_NAM,-
            1430         FAC = <PUT,GET>,-
            1430         MRS = 2*TEXTB_SIZE,-
            1430         ORG = SEQ
            1480
            1480  SE_NAM: $NAM-                                    ; Used for copy of slave's SYS$ERROR
            1480         RSS = NAM$C_MAXRSS,-
            1480         RSA = SE_FILESPEC
            14E0
            14E0  SE_RAB:                                          ; Used for copy of slave's SYS$ERROR
            14E0         $RAB-
            14E0         FAB = SE_FAB
            1524
            1524  SE_FILESPEC:                                     ; Used for copy of slave's SYS$ERROR
00001623    1524         .BLKB   NAM$C_MAXRSS
            1623
            1623  RF_FAB:                                          ; Used to create files on cluster nodes
            1623         $FAB-
            1623         FNA = RF_FILESPEC,-
            1623         FOP = <SUP>,-
            1623         FAC = <PUT,GET>,-
            1623         NAM = RF_NAM,-
            1623         SHR = <PUT,GET,UPI>,-
            1623         MRS = TEXTB_SIZE,-
            1623         ORG = SEQ
            1673
            1673  RF_NAM:                                          ; Used to create files on cluster nodes
            1673         $NAM-
            1673         RSS = NAM$C_MAXRSS,-
            1673         RSA = RESULT_FILESPEC
            16D3
            16D3  RF_RAB:                                          ; Used to create files on cluster nodes
            16D3         $RAB-
            16D3         FAB = RF_FAB,-
            16D3         ROP = <NLK>,-
            16D3         RSZ = TEXTB_SIZE,-
            16D3         RBF = BUFFER,-
            16D3         USZ = TEXTB_SIZE,-
            16D3         UBF = BUFFER
            1717
            1717  RF_FILESPEC_DESC:                                ; String descriptor for error messages
0000171B    1717         .BLKW   2
0000171F'   171B         .ADDRESS RF_FILESPEC
            171F
            171F  RF_FILESPEC:                                     ; Holds filespecs for test files
0000181E    171F         .BLKB   NAM$C_MAXRSS
            181E
            181E  RESULT_FILESPEC:                                 ; Receives resultant test file filespec
0000191D    181E         .BLKB   NAM$C_MAXRSS
```

```
                             191D        .SBTTL  Main Program
                         00000000        .PSECT  _UETP$CODE,EXE,NOWRT,PIC,SHR,PAGE
                             0000
                             0000        .DEFAULT DISPLACEMENT,WORD
                             0000
                             0000   ;+
                             0000   ;    The UETP Cluster Integration test will test the cluster functions
                             0000   ;    available to typical user applications.  It relies very heavily
                             0000   ;    on DECnet.
                             0000   ;
                             0000   ;    The node from which the test is originally run is called the master
                             0000   ;    node.  VMS nodes in the cluster which run the test at the request of
                             0000   ;    the master node are called slave nodes.  The main flow of testing is:
                             0000   ;            If we are in a cluster then
                             0000   ;                If we are the master process then
                             0000   ;                    Get a list of VAX cluster nodes. Warn each of testing
                             0000   ;                    Initiate a DECnet link to each VAX cluster node
                             0000   ;                    Start a slave task on each such node
                             0000   ;                    Have each node take out a lock (no deadlock)
                             0000   ;                    Have each node take out another lock (to check deadlock)
                             0000   ;                    Check that file access works to all cluster nodes
                             0000   ;                    Terminate slave processes
                             0000   ;                    Send an end of testing message to all cluster consoles
                             0000   ;                Else
                             0000   ;                    Complete the DECnet link to the master process
                             0000   ;                    Take out a lock (no deadlock)
                             0000   ;                    Take out another lock (in order to check deadlock)
                             0000   ;                    Wait to be told what to do next
                             0000   ;            Exit the test
                             0000   ;-
                             0000
                     0000    0000   .ENTRY UETCLIG00,^M<>                     ; Entry mask
                             0002
        6D   1C15'CF   DE    0002        MOVAL   SSERROR,(FP)                 ; Declare exception handler
                             0007        $SETSFM_S ENBFLG = #1                ; Enable system service failure mode
                             0010        $TRNLOG_S LOGNAM = SYS$NET,-         ; Are we a slave or a master process?
                             0010                  RSLBUF = FAO_BUF
        50   0000'8F   B1    0027        CMPW    #SS$_NOTRAN,R0               ; If SYS$NET is undefined...
                  23   13    002C        BEQL    10$                          ; ...then we're a master process
        0024'CF   02   C8    002E        BISL2   #CLIG_M_SLAVE,FLAGS          ; Otherwise, mark us as a slave...
                             0033        $CREATE FAB = SE_FAB,-               ; ...and set up our copy of SYS$ERROR
                             0033                ERR = RMS_ERROR
                             0042        $CONNECT RAB = SE_RAB,-
                             0042                ERR = RMS_ERROR
                             0051   10$:
                             0051        $DCLEXH_S DESBLK = EXIT_DESC         ; Declare an exit handler
                             005C
                             005C        $GETSYI_S ITMLST = MYNODE_ITMLST     ; Get my node's node name
    0042'CF   06   00   3A   0071        LOCC    #0,#NODE_LENGTH,SCSNODE      ; Ensure that...
61   50   20   00 8F   00   2C   0077        MOVC5   #0,#0,#^X/ /,R0,(R1)         ; ...the name is blank filled
                             007E
                             007E        $GETJPI_S ITMLST = MYPROC_ITMLST     ; Find out my process name
        56   009D'CF   7E    0093        MOVAQ   UETCLIG,R6                   ; Define a new one...
        57   0042'CF   9E    0098        MOVAB   SCSNODE,R7                   ; ...assuming we are a slave...
     0A 0024'CF   01   E0    009D        BBS     #CLIG_V_SLAVE,FLAGS,20$
        56   0000'CF   7E    00A3        MOVAQ   PROCESS_NAME,R6              ; ...but different...
        57   00B5'CF   9E    00A8        MOVAB   MASTER+8,R7                  ; ...if we're master
```

```
                                 00AD      20$:
       58   0069'CF       9E     00AD              MOVAB   NEWNAM,R8                   ; We'll use the new one...
  68    08   A6   66      28     00B2              MOVC3   (R6),8(R6),(R8)             ; ...
       63    67   06      28     00B7              MOVC3   #NODE_LENGTH,(R7),(R3)      ; ...
 0061'CF     53   58      A3     00BB              SUBW3   R8,R3,NEWNAM_DESC           ; ...
                                 00C1              $SETSFM_S ENBFLG = #0
                                 00CA              $SETPRN_S PRCNAM = NEWNAM_DESC    ; ...while running this test
                                 00D5              $SETSFM_S ENBFLG = #1
 000C'CF     0061'CF      7E     00DE              MOVAQ    NEWNAM_DESC,CLIG_ANNOUNCE+12 ; Use process name in sentinel msgs
                                 00E5              $PUTMSG_S MSGVEC = CLIG_ANNOUNCE,- ; Give a beginning message
                                 00E5                      ACTRTN = SE_COPY
       0024'CF    08      C8     00F8              BISL2   #CLIG_M_BEGINMSG,FLAGS    ; Set flag so we don't print it again
                                 00FD
                                 00FD              $TRNLOG_S LOGNAM = MODE-          ; See if the user wants tracing info
                                 00FD                      RSLBUF = FAO_BUF
       50    0000'8F      B1     0114              CMPW    #SS$_NOTRAN,R0            ; If MODE logical name defined...
                   25     13     0119              BEQL    30$
 005C'DF     0058'CF      39     011B              MATCHC  DUMP,@DUMP+4,-            ; ...as "DUMP"...
 0CC4'CF     021A 8F             0122                      #2*TEXTB_SIZE,BUFFER
                   16     12     0128              BNEQ    30$
       0024'CF    01      C8     012A              BISL2   #CLIG_M_DEBUG,FLAGS       ; ...remember that user wants trace info
 0FF3'CF     0A09'CF      7D     012F              MOVQ    DEBUG_INTRO_MSG,DEBUG_PTR ; Warn the user...
                 1A70     30     0136              BSBW    GIVE_DEBUG_MSG            ; ...if there will be extra messages
 0FF7'CF     0FFB'CF      DE     0139              MOVAL   DEBUG_BUFFER,DEBUG_PTR+4 ; Reset standard pointer
                                 0140      30$:
                                 0140
                                 0140              $GETDVIW_S DEVNAM = SYS$INPUT,-    ; Get the name of the device...
                                 0140                      ITMLST = INPUT_ITMLST,- ; ...which may abort the test
                                 0140                      EFN    = #SS_SYNCH_EFN,-
                                 0140                      IOSB   = QUAD_STATUS
        49 002C'CF        E9     015C              BLBC    QUAD_STATUS,40$          ; Avoid CTRL/C handler if any error
     43 003E'CF    00'    E1     0161              BBC     S^#DEV$V_TRM,DEVCHAR,40$ ; BR if SYS$INPUT is NOT a terminal
                                 0167              $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C AST handler
                                 0167                      CHAN   = TTCHAN
                                 0178              $QIOW_S CHAN   = TTCHAN,-        ; Enable CTRL/C ASTs
                                 0178                      FUNC   = #IO$_SETMODE!IO$M_CTRLCAST,-
                                 0178                      P1     = CCASTHAND
                                 0199              $PUTMSG_S MSGVEC = ABORTC_MSG_PTR ; Tell user how to abort gracefully
                                 01AA      40$:
                                 01AA
                                 01AA              IFCLSTR 50$                      ; BR if we're a cluster member...
                                 01B2              $PUTMSG_S MSGVEC = LONELY_MSG_PTR,- ; ...else say there's no testing
                                 01B2                      ACTRTN = SE_COPY
                   29     11     01C5              BRB     70$
                                 01C7      50$:
     17 0024'CF    01     E0     01C7              BBS     #CLIG_V_SLAVE,FLAGS,60$ ; BR if we are a slave process
                 002D     30     01CD              BSBW    ANNOUNCE_US              ; Let systems know of our test
                 00FF     30     01D0              BSBW    GET_NODES                ; Collect nodes in cluster, start DECnet
                 0300     30     01D3              BSBW    START_TALKING            ; Say "Hi" to the other nodes
                 03CA     30     01D6              BSBW    CHECK_LOCKS              ; See if locks are cluster visible
                 05DE     30     01D9              BSBW    CHECK_DEADLOCK           ; See if deadlock detection works
                 0BD3     30     01DC              BSBW    FILE_ACCESS              ; See if we can get to cluster files
                 132B     30     01DF              BSBW    WIND_DOWN                ; Terminate slaves and clean up
                   0C     11     01E2              BRB     70$                      ; Exit successfully
                                 01E4      60$:
                 035A     30     01E4              BSBW    SET_UP_SLAVE             ; Set up the DECnet link to master
                 04EF     30     01E7              BSBW    TAKE_OUT_LOCK            ; See if locks work in the cluster
```

F 8

```
            09AA   30  01EA   872           BSBW    GET_DEADLOCK            ; Participate in a deadlock
            10C2   30  01ED   873           BSBW    SHARE_ACCESS            ; Access a file in use by master process
                       01F0   874 70$:
                       01F0   875           $EXIT_S CODE = -                ; Exit with a successful status
                       01F0   876                   #SS$_NORMAL!STS$M_INHIB_MSG
```

UE
V0

```
                         01FD        .SBTTL  ANNOUNCE_US - Let Systems Know of Our Test
                         01FD   ;++
                         01FD   ; FUNCTIONAL DESCRIPTION:
                         01FD   ;       Get the names of all the nodes in the cluster.
                         01FD   ;       For record keeping purposes, it's a good idea to let other systems in
                         01FD   ;       the cluster know that we're about to start testing.  Put a message to
                         01FD   ;       the operator's console on each VAX node, itself a test of $BRKTHRU.
                         01FD   ;
                         01FD   ; IMPLICIT INPUTS:
                         01FD   ;       VMS's list of cluster (VMS and non-VMS both) nodes
                         01FD   ;
                         01FD   ; IMPLICIT OUTPUTS:
                         01FD   ;       Copy of our node's view of the cluster
                         01FD   ;
                         01FD   ; SIDE EFFECTS:
                         01FD   ;       Message to all console terminals in the cluster.
                         01FD   ;       P0 space expanded to include output from UETP$CLSIODB.
                         01FD   ;
                         01FD   ;--
                         01FD
                         01FD   ANNOUNCE_US:
                         01FD           $CMKRNL_S ROUTIN = UETP$CLSIODB,- ; Form a list of other cluster...
                         01FD                   ARGLST = CLSIODB_ARGS     ; ...nodes and SCS peripherals
         24 50    E8     020C           BLBS    R0,10$                    ; BR if the list was formed correctly
            50    DD     020F           PUSHL   R0                        ; Save the error status
  1BC3'CF   01    FB     0211           CALLS   #1,STATUS_TO_TEXT         ; Get the text for it
     0EDE'CF      DF     0216           PUSHAL  STATUS_PTR                ; Explain what went wrong
            01    DD     021A           PUSHL   #1
  00741134  8F    DD     021C           PUSHL   #UETP$_TEXT!STS$K_SEVERE
     02F3'CF      DF     0222           PUSHAL  CLSIODB_FAIL
            01    DD     0226           PUSHL   #1
  00741134  8F    DD     0228           PUSHL   #UETP$_TEXT!STS$K_SEVERE
            06    DD     022E           PUSHL   #6
          1BCD    31     0230           BRW     ERROR_EXIT                ; We can't continue
                         0233   10$:
  50   0042'CF    DE     0233           MOVAL   SCSNODE,R0
                         0238           $FAO_S  CTRSTR = WARN_OF_TESTING,-
                         0238                   OUTLEN = BUFFER_PTR,-
                         0238                   OUTBUF = FAO_BUF,-
                         0238                   P1     = #NODE_LENGTH,-
                         0238                   P2     = R0,-
                         0238                   P3     = #0
                         0251           $BRKTHRUW_S -                     ; Warn other nodes by a console message
                         0251                   MSGBUF = BUFFER_PTR,-
                         0251                   EFN    = #SS_SYNCH_EFN,-
                         0251                   SENDTO = OPA0,-
                         0251                   SNDTYP = #BRK$C_DEVICE,-
                         0251                   FLAGS  = #BRK$M_CLUSTER,-
                         0251                   TIMOUT = #BRKTHRU_TIMOUT,-
                         0251                   IOSB   = QUAD_STATUS
    0A 002C'CF    E9     0276           BLBC    QUAD_STATUS,20$           ; BR if there was any error in sending
       0030'CF    A1     027B           ADDW3   QUAD_STATUS+4,-           ; Did all nodes see the warning?
  51   0032'CF           027F                   QUAD_STATUS+6,R1
            4C    13     0283           BEQL    30$                       ; BR if so - all OK
                         0285   20$:
  7E   002C'CF    3C     0285           MOVZWL  QUAD_STATUS,-(SP)         ; Get the text...
  1BC3'CF   01    FB     028A           CALLS   #1,STATUS_TO_TEXT         ; ...associated with any error
```

```
                                                                       H  8
UETCLIG00                                     VAX/VMS UETP Cluster Integration Test     16-SEP-1984 00:19:09  VAX/VMS Macro V04-00          Page  24
V04-000                                       ANNOUNCE_US - Let Systems Know of Our Te   6-SEP-1984 10:00:47  [UETPSY.SRC]UETCLIG00.MAR;1        (7)

                          51   0030'CF   3C  028F    935           MOVZWL  QUAD_STATUS+4,R1
                          52   0032'CF   3C  0294    936           MOVZWL  QUAD_STATUS+6,R2
                                             0299    937           $FAO_S  CTRSTR = BRKTHRU_ERRORS,- ; Form a message
                                             0299    938                   OUTLEN = BUFFER_PTR,-
                                             0299    939                   OUTBUF = FAO_BUF,-
                                             0299    940                   P1     = R1,-
                                             0299    941                   P2     = R2
                               0EDE'CF   DF  02B0    942           PUSHAL  STATUS_PTR
                                    01   DD  02B4    943           PUSHL   #1
                           00741132 8F   DD  02B6    944           PUSHL   #UETP$_TEXT!STS$K_ERROR
                               0CBC'CF   DF  02BC    945           PUSHAL  BUFFER_PTR
                           000F0001 8F   DD  02C0    946           PUSHL   #^XF0001
                           00741132 8F   DD  02C6    947           PUSHL   #UETP$_TEXT!STS$K_ERROR
                          1DAD'CF   06   FB  02CC    948           CALLS   #6,ERROR_SIGNAL          ; Let users know of any problems
                                             02D1    949 30$:
                                         05  02D1    950           RSB
```

```
                              02D2            .SBTTL  GET_NODES - Collect the DECnet/VAX Nodes in Our Cluster
                              02D2    ;++
                              02D2    ; FUNCTIONAL DESCRIPTION:
                              02D2    ;       Form descriptors to the names of the VAX/VMS nodes.  See if we're
                              02D2    ;       running DECnet to those nodes by establishing a link and starting up a
                              02D2    ;       task on the node.  In order that we may recover from not being able
                              02D2    ;       to DECnet to a node or nodes, turn off System Service failure mode
                              02D2    ;       and explicitly check for errors.
                              02D2    ;
                              02D2    ; IMPLICIT INPUTS:
                              02D2    ;       The list of cluster nodes from UETP$CLSIODB
                              02D2    ;
                              02D2    ; IMPLICIT OUTPUTS:
                              02D2    ;       NODE_CHANS has a channel number for all those nodes to which we were
                              02D2    ;               able to establish a DECnet link.
                              02D2    ;       NODE_NAMES has a descriptor to all the names of the VMS nodes.
                              02D2    ;
                              02D2    ; SIDE EFFECTS:
                              02D2    ;       DECnet links to and remote tasks on VMS cluster nodes.
                              02D2    ;       Warning messages if we were unable to establish a link to such a node.
                              02D2    ;
                              02D2    ;--
                              02D2
                              02D2    GET_NODES:
     56   00A2'CF       D0    02D2            MOVL    CLSPTR,R6                 ; Used to loop through system records
     57   00AA'CF       3E    02D7            MOVAW   NODE_CHANS,R7             ; Used to loop through channel words
     58   02AA'CF       7E    02DC            MOVAQ   NODE_NAMES,R8             ; Used to loop through name descriptors
                              02E1    10$:
              01        91    02E1            CMPB    #UID$K_SID_RTYPE,-        ; Is this a system block record?
           06 A6              02E3                    UIDGNRC$B_TYPE(R6)
              11        13    02E5            BEQL    20$                       ; BR if it is
         032C'CF        DF    02E7            PUSHAL  CLSIODB_SCREWEY           ; Die noisily if it is isn't
              01        DD    02EB            PUSHL   #1
     00741134 8F        DD    02ED            PUSHL   #UETP$_TEXT!STS$K_SEVERE
              03        DD    02F3            PUSHL   #3
            1B08        31    02F5            BRW     ERROR_EXIT
                              02F8    20$:
  11 A6   0099'CF       D1    02F8            CMPL    VMS,UIDSID$T_SWTYPE(R6) ; Is this a VAX/VMS node?
                              02FE            BNEQW   60$                       ; BR if it is not
           07 A6        D5    0303            TSTL    UIDSID$L_PBFL(R6)         ; Have we any path to the node?
                              0306            BEQLW   60$                       ; BR if not - we can't test it
     68    31 A6        9B    030B            MOVZBW  UIDSID$T_NODENAME(R6),(R8) ; Save the length of the name...
           32 A6        DE    030F            MOVAL   UIDSID$T_NODENAME+1(R6),-  ; ...and its address
           04 A8              0312                    4(R8)
                              0314            $SETSFM_S ENBFLG = #0             ; Turn off SS errors...
                              031D            $GETSYIW_S EFN    = #SS_SYNCH_EFN,- ; ...while checking to see...
                              031D                    IOSB      = QUAD_STATUS,-   ; ...if this node is in our cluster
                              031D                    ITMLST    = OTHERNODE_ITMLST,-
                              031D                    NODENAME = (R8)
     52    50           D0    0334            MOVL    R0,R2                     ; Preserve the return status...
                              0337            $SETSFM_S ENBFLG = #1             ; ...while resuming SS error checking
         0A 52          E9    0340            BLBC    R2,30$                    ; BR if it is not a member
    05 002C'CF          E9    0343            BLBC    QUAD_STATUS,30$           ; BR if it is not
    29 0090'CF          E8    0348            BLBS    CLUSTER_MEMBER,40$        ; BR if it finally is
                              034D    30$:    $FAO_S  CTRSTR = REBEL_MSG,-      ; Tell user that we can't test it
                              034D                    OUTLEN = BUFFER_PTR,-
                              034D                    OUTBUF = FAO_BUF,-
```

```
                                   034D                  P1     = R8
                                   0362          $PUTMSG_S MSGVEC = REBEL_MSG_PTR
                    0083    31     0373          BRW     60$                   ; "Next" item will overwrite this one
                                   0376    40$:
   0CC4'CF    04 B8   68    28     0376          MOVC3   (R8),@4(R8),BUFFER    ; Concatenate the node name with the...
63   0075'DF   0071'CF    28     037D          MOVC3   TASK,@TASK+4,(R3)     ; ...rest of the DECnet target string
0CBC'CF   0071'CF    68    A1     0385          ADDW3   (R8),TASK,BUFFER_PTR  ; Form a descriptor for the string
                                   038D          $SETSFM_S ENBFLG = #0         ; Turn off SS errors...
                                   0396          $ASSIGN_S DEVNAM = BUFFER_PTR,- ; ...while getting a DECnet link...
                                   0396                  CHAN   = (R7)
                 52    50    D0     03A5          MOVL    R0,R2                 ; Preserve the return status...
                                   03A8          $SETSFM_S ENBFLG = #1         ; ...while restoring error handling
               41 52    E8          03B1          BLBS    R2,50$                ; ...so we don't bomb out...
                       52    DD     03B4          PUSHL   R2                    ; ...if we should get an error
           1BC3'CF    01    FB     03B6          CALLS   #1,STATUS_TO_TEXT     ; Get the text for the error code...
                                   03BB          $FAO_S  CTRSTR = LINK_FAILED,- ; ...and an explanatory message...
                                   03BB                  OUTLEN = BUFFER_PTR,-
                                   03BB                  OUTBUF = FAO_BUF,-
                                   03BB                  P1     = R8,-
                                   03BB                  P2     = R8
              0EDE'CF       DF     03D2          PUSHAL  STATUS_PTR
                    01       DD     03D6          PUSHL   #1
          00741132 8F       DD     03D8          PUSHL   #UETP$_TEXT!STS$K_ERROR
              0CBC'CF       DF     03DE          PUSHAL  BUFFER_PTR
          000F0001 8F       DD     03E2          PUSHL   #^XF0001
          00741132 8F       DD     03E8          PUSHL   #UETP$_TEXT!STS$K_ERROR
           1DAD'CF    06    FB     03EE          CALLS   #6,ERROR_SIGNAL       ; ...and signal the error
                    04       11     03F3          BRB     60$                   ; Let "next" node overwrite this one
                                   03F5    50$:
                    87       B5     03F5          TSTW    (R7)+                 ; Point to the next space for channel
                    88       73     03F7          TSTD    (R8)+                 ; Point to the next space for name desc
                                   03F9    60$:
                 56    66    D0     03F9          MOVL    UIDSID$A_FLINK(R6),R6 ; Point to the next possible SID record
                                   03FC          BNEQW   10$                   ; Loop for another node if there's one
```

```
                             0401        ;
                             0401        ; Set up an $FAOL PRMLST so we can tell the world which nodes we're testing.
                             0401        ;
         57   00AA'CF   3E   0401           MOVAW   NODE_CHANS,R7              ; Used to loop through channel words
         58   02AA'CF   7E   0406           MOVAQ   NODE_NAMES,R8              ; Used to loop through name descriptors
              59   01   CE   040B           MNEGL   #1,R9                      ; This will count items to print
                             040E                                              ; Sleaze: Last COMMASPACE not printed!
    56   045B'CF   06   A3   040E           SUBW3   #6,NODE_LIST_MSG,R6        ; Initialize line length
    5E   00000EF1  8F   C2   0414           SUBL2   #<4+4+2+4+1>*MAX_NODES,SP ; We need a throwaway data str...
              5B   5E   D0   041B           MOVL    SP,R11                     ; ...to store some throwaway data
    5E   000003FC  8F   C2   041E           SUBL2   #4*MAX_NODES,SP            ; Preallocate a worst-case amount...
              5A   5E   D0   0425           MOVL    SP,R10                     ; ...of space for $FAOL PRMLST
                             0428   70$:
                   87   B5   0428           TSTW    (R7)+                      ; Will we try testing another node?
                   3B   13   042A           BEQL    90$                        ; BR if we're at the end of the list
         0F   0050 8F   3D   042C           ACBW    #80,#<NODE_LENGTH+2+2+4+1>,- ; BR if this node and version...
              000A 56        0431                   R6,80$                     ;   ...won't wrap the line
         8A   0492'CF   7E   0434           MOVAQ   CRLFTAB,(R10)+             ; Wrap the line neatly
              56   08   B0   0439           MOVW    #8,R6                      ; Reinitialize the line length
                   59   D6   043C           INCL    R9                         ; Count the line wrap as item to print
                             043E   80$:
              8A   68   7E   043E           MOVAQ   (R8),(R10)+                ; Put the node desc in our PRMLST
              8A   5B   D0   0441           MOVL    R11,(R10)+                 ; Save a pointer...
              8B   07   D0   0444           MOVL    #<2+4+1>,(R11)+            ; ...to a descriptor...
         8B   04   AB   DE   0447           MOVAL   4(R11),(R11)+              ; ...in our throwaway data structure...
       8B   2820   8F   B0   044B           MOVW    #^A/ (/,(R11)+             ; ...that's used to display...
         50   04   A8   D0   0450           MOVL    4(R8),R0                   ; ...
         8B   E3   A0   D0   0454           MOVL    <UIDSID$T_SWVERS--         ; ...the software version...
                             0458                   UIDSID$T_NODENAME-1>(R0),(R11)+
              8B   29   90   0458           MOVB    #^A/)/,(R11)+              ; ...running on this node
         8A   0488'CF   7E   045B           MOVAQ   COMMASPACE,(R10)+          ; Separate successive nodes
              59   03   C0   0460           ADDL2   #3,R9                      ; Count items on the PRMLST
                             0463
                   88   73   0463           TSTD    (R8)+                      ; Point to the next possible node desc
                   C1   11   0465           BRB     70$                        ; Loop for more nodes
                             0467   90$:
                   59   D5   0467           TSTL    R9                         ; Were any nodes to be tested?
                   13   14   0469           BGTR    100$                       ; BR if there were
                             046B           $PUTMSG_S MSGVEC = NO_NODE_MSG_PTR ; Let the world know if there weren't
                   50   11   047C           BRB     110$                       ; Use common exit
                             047E   100$:
                             047E           $TRNLOG_S LOGNAM = REPORT,-        ; See if the user wants misc info
                             047E                   RSLBUF = FAO_BUF
0CC4'CF   0047'CF   003F'CF   29   0495     CMPC3   SHORT,SHORT+8,BUFFER       ; If "short" report was requested...
                   2D   13   049F           BEQL    110$                       ; ...then BR to omit the message
                   59   DD   04A1           PUSHL   R9                         ; Save parameter count
              5B   5E   D0   04A3           MOVL    SP,R11                     ; Save the pointer to the PRMLST
                             04A6           $FAOL_S CTRSTR = NODE_LIST_MSG,- ; Form a message with node names
                             04A6                   OUTLEN = BUFFER_PTR,-
                             04A6                   OUTBUF = FAO_BUF,-
                             04A6                   PRMLST = (R11)
                   01   BA   04BB           POPR    #^M<R0>                    ; Remove parameter count
                             04BD           $PUTMSG_S -                        ; List the node names for the user
                             04BD                   MSGVEC = NODE_LIST_MSG_PTR
                             04CE   110$:
    5E   000012ED  8F   C0   04CE           ADDL2   #<4+4+2+4+1+4>*MAX_NODES,SP ; Clean up the stack
                   05        04D5           RSB                                ; We're done
```

```
                                04D6              .SBTTL  START_TALKING - Start Communications Between Master and Slaves
                                04D6        ;++
                                04D6        ; FUNCTIONAL DESCRIPTION:
                                04D6        ;       Start communicating with the tasks established by GET_NODES.  (Those
                                04D6        ;       tasks will be running this same image, but take a different execution
                                04D6        ;       path because there will be a translation for the logical name SYS$NET.)
                                04D6        ;       We start communicating with each "slave" by exchanging greetings.
                                04D6        ;
                                04D6        ; IMPLICIT INPUTS:
                                04D6        ;       NODE_CHAN list of channels on which we have DECnet links.
                                04D6        ;       NODE_NAMES list of pointers to descriptors of node names with which
                                04D6        ;               we've established a link.
                                04D6        ;
                                04D6        ; IMPLICIT OUTPUTS:
                                04D6        ;       NONE
                                04D6        ;
                                04D6        ; SIDE EFFECTS:
                                04D6        ;       Messages to tasks on those nodes.
                                04D6        ;
                                04D6        ;--
                                04D6
                                04D6        START_TALKING:
              57   00AA'CF   3E 04D6                MOVAW   NODE_CHANS,R7                 ; Used to loop through DECnet channels
              58   02AA'CF   7E 04DB                MOVAQ   NODE_NAMES,R8                 ; Used to loop through node name descs
              59   0DB2'CF   DE 04E0                MOVAL   HELLO_MSG,R9                  ; Set up convenience registers...
              5A   0DB9'CF   DE 04E5                MOVAL   IMOK_MSG,R10                  ; ...
     0AA2'CF    02 A9    69  28 04EA                MOVC3   (R9),2(R9),MESSAGE_BUFFER     ; Set up msg to tell each slave...
          63   0042'CF   06  28 04F1                MOVC3   #NODE_LENGTH,SCSNODE,(R3)     ; ...the name of the master node
                                04F7        10$:
                         67  B5 04F7                TSTW    (R7)                          ; Have we another channel?
                         01  12 04F9                BNEQ    20$                           ; BR if so - send a message
                         05     04FB                RSB                                   ; Return if not
                                04FC        20$:
                   7E    67  3C 04FC                MOVZWL  (R7),-(SP)                    ; Set up the channel...
                         58  DD 04FF                PUSHL   R8                            ; ...the node name...
                         59  DD 0501                PUSHL   R9                            ; ...and our message name
              1922'CF    03  FB 0503                CALLS   #3,MASTER_WRITE               ; Say "HI!" to the next node
                   30    50  E9 0508                BLBC    R0,40$                        ; Skip the rest if this node died
                   7E    67  3C 050B                MOVZWL  (R7),-(SP)                    ; Set up the channel...
                         58  DD 050E                PUSHL   R8                            ; ...the node name...
                         5A  DD 0510                PUSHL   R10                           ; ...and our message name
              19B0'CF    03  FB 0512                CALLS   #3,MASTER_READ                ; See if this node knows us
                   21    50  E9 0517                BLBC    R0,40$                        ; Skip the rest if no reply
     0CC4'CF    02 AA    6A  29 051A                CMPC3   (R10),2(R10),BUFFER           ; Did we get the reply we wanted?
                         07  12 0521                BNEQ    30$                           ; BR if not
          63    04 B8    68  29 0523                CMPC3   (R8),@4(R8),(R3)              ; Was reply from the node we wanted?
                         11  13 0528                BEQL    40$                           ; BR if it was
                                052A        30$:
              0999'CF        DF 052A                PUSHAL  EXCLUDE_MSG                   ; Complain that we got back trash
                         58  DD 052E                PUSHL   R8
                         5A  DD 0530                PUSHL   R10
              1B47'CF    03  FB 0532                CALLS   #3,GARBLED_TRANS
                02 A8    02  A8 0537                BISW2   #CLIG_M_DEADNODE,2(R8)        ; Indicate that we're done with node
                                053B        40$:
                         87  B5 053B                TSTW    (R7)+                         ; Point to the next possible channel
                         88  73 053D                TSTD    (R8)+                         ; Point to the next possible name desc
                         B6  11 053F                BRB     10$                           ; Loop to say hi to the next one
```

```
                                    0541          .SBTTL  SET_UP_SLAVE - Complete DECnet Link to Master
                                    0541   ;++
                                    0541   ; FUNCTIONAL DESCRIPTION:
                                    0541   ;       We've been started up as a DECnet task.  Complete the link to the
                                    0541   ;       process which started us.
                                    0541   ;
                                    0541   ; IMPLICIT INPUTS:
                                    0541   ;       SYS$NET logical name is defined.
                                    0541   ;
                                    0541   ; IMPLICIT OUTPUTS:
                                    0541   ;       NODE_CHANS gets DECnet channel number
                                    0541   ;
                                    0541   ; SIDE EFFECTS:
                                    0541   ;       DECnet link is completed.
                                    0541   ;
                                    0541   ;--
                                    0541
                                    0541   SET_UP_SLAVE:
            59   0DB2'CF      DE    0541           MOVAL   HELLO_MSG,R9            ; Set up convenience registers...
            5A   0DB9'CF      DE    0546           MOVAL   IMOK_MSG,R10            ; ...
                                    054B           $ASSIGN_S DEVNAM = SYS$NET,-    ; Complete DECnet link to master process
                                    054B                   CHAN   = NODE_CHANS
                        59    DD    055C           PUSHL   R9                      ; Define the type of message we want
               16D0'CF  01    FB    055E           CALLS   #1,SLAVE_READ           ; Get the master node's "HELLO" message
  0AA2'CF    02 A9      69    29    0563           CMPC3   (R9),2(R9),MESSAGE_BUFFER ; What does the message say?
                        1C    13    056A           BEQL    10$                     ; BR if it says "HELLO"
                   00BB'CF    DF    056C           PUSHAL  NULL                    ; Otherwise,...
                   00AD'CF    DF    0570           PUSHAL  MASTER
                        59    DD    0574           PUSHL   R9
               1B47'CF  03    FB    0576           CALLS   #3,GARBLED_TRANS        ; ...signal the error
                                    057B           $EXIT_S CODE = #UETP$_ABENDD!STS$K_ERROR!STS$M_INHIB_MSG
                                    0588   10$:
                   63   06    28    0588           MOVC3   #NODE_LENGTH,(R3),-     ; Save the master node's name
                   009C'CF          058B                   MASTER_NODE
                 02 AA  6A    28    058E           MOVC3   (R10),2(R10),-          ; Set up msg telling master node...
                   0AA2'CF          0592                   MESSAGE_BUFFER
                        06    28    0595           MOVC3   #NODE_LENGTH,-          ; ...that I'm an OK node
            63   0042'CF            0597                   SCSNODE,(R3)
                        5A    DD    059B           PUSHL   R10                     ; Define the type of message we want
               1769'CF  01    FB    059D           CALLS   #1,SLAVE_WRITE          ; Tell the master node that I'm OK
                              05    05A2           RSB
```

```
                                      05A3        .SBTTL  CHECK_LOCKS - See If Locks are Cluster Visible
                                      05A3  ;++
                                      05A3  ; FUNCTIONAL DESCRIPTION:
                                      05A3  ;       Take out a lock and see that it's visible from the master node.  To
                                      05A3  ;       allow for the possibility of the test being run simultaneously from
                                      05A3  ;       mode than one node in a cluster, choose a lock name that we can
                                      05A3  ;       guarantee will be unique amongst cooperating tests.  Lock names will
                                      05A3  ;       be an identifying string, concatenated with the master node name
                                      05A3  ;       (already known to slave nodes), concatenated with the name of the node
                                      05A3  ;       taking the lock, concatenated with a string supplied by the master.
                                      05A3  ;       For this step, the string will repeat the name of the node taking the
                                      05A3  ;       lock.  (See the deadlock detection section for a later use of this
                                      05A3  ;       lock.)  Check that the lock is visible.  Take out a corresponding
                                      05A3  ;       lock for the master node.
                                      05A3  ;
                                      05A3  ; IMPLICIT INPUTS:
                                      05A3  ;       NONE
                                      05A3  ;
                                      05A3  ; IMPLICIT OUTPUTS:
                                      05A3  ;       NONE
                                      05A3  ;
                                      05A3  ; SIDE EFFECTS:
                                      05A3  ;       A set of locks, one for each slave process.  The resource names
                                      05A3  ;               have the form, "id-string_master-node_slave-node_slave-node",
                                      05A3  ;               where all node names are assumed to be NODE_LENGTH characters.
                                      05A3  ;
                                      05A3  ;--
                                      05A3
                                      05A3  CHECK_LOCKS:
           57    00AA'CF      3E      05A3        MOVAW   NODE_CHANS,R7             ; Used to loop through DECnet channels
           58    02AA'CF      7E      05A8        MOVAQ   NODE_NAMES,R8             ; Used to loop through node name descs
           59    0DBF'CF      DE      05AD        MOVAL   TAKELOCK_MSG,R9           ; Set up convenience registers...
           5A    0DC9'CF      DE      05B2        MOVAL   GOTLOCK_MSG,R10           ; ...
      00    02 A9      69     2C      05B7        MOVC5   (R9),2(R9),#0,-           ; Set up msg telling slaves...
                 010D 8F              05BC                #TEXTB_SIZE,-             ; ...to take out a lock
                 0AA2'CF              05BF                MESSAGE_BUFFER
                                      05C2  10$:
                       67     B5      05C2        TSTW    (R7)                      ; Have we another channel?
                       01     12      05C4        BNEQ    20$                       ; BR if so - send a message
                       05             05C6        RSB                               ; Return if not
                                      05C7  20$:
                                      05C7        BBSW    #CLIG_V_DEADNODE,2(R8),60$ ; BR to next node if this one is dead
                 50    69     3C      05CF        MOVZWL  (R9),R0                   ; Append node name to the message...
         50   0AA2'CF40       9E      05D2        MOVAB   MESSAGE_BUFFER[R0],R0     ; ...
      60    04 B8      06     28      05D8        MOVC3   #NODE_LENGTH,a4(R8),(R0)  ; ...so slave knows resource to lock
           7E          67     3C      05DD        MOVZWL  (R7),-(SP)                ; Set up the channel...
                       58     DD      05E0        PUSHL   R8                        ; ...the node name...
                       59     DD      05E2        PUSHL   R9                        ; ...and our message name
         1922'CF       03     FB      05E4        CALLS   #3,MASTER_WRITE           ; Tell this node to get a lock
                                      05E9        BLBCW   R0,60$                    ; Skip the rest if this node died
           7E          67     3C      05EF        MOVZWL  (R7),-(SP)                ; Set up the channel...
                       58     DD      05F2        PUSHL   R8                        ; ...the node name...
                       5A     DD      05F4        PUSHL   R10                       ; ...and our message name
         19B0'CF       03     FB      05F6        CALLS   #3,MASTER_READ            ; See if this node got the lock
                                      05FB        BLBCW   R0,60$                    ; Error in sending, skip the rest
 0CC4'CF    02 AA      6A     29      0601        CMPC3   (R10),2(R10),BUFFER       ; Did we get the reply we wanted?
                       07     12      0608        BNEQ    30$                       ; BR if not
      63    04 B8    68    29  060A           CMPC3   (R8),@4(R8),(R3)            ; Was reply from the node we wanted?
                     14    13  060F           BEQL    40$                         ; BR if it was
                               0611      30$:
                0999'CF    DF  0611           PUSHAL  EXCLUDE_MSG                 ; Complain that we got back trash
                     58    DD  0615           PUSHL   R8
                     5A    DD  0617           PUSHL   R10
            1B47'CF  03    FB  0619           CALLS   #3,GARBLED_TRANS
              02 A8  02    A8  061E           BISW2   #CLIG_M_DEADNODE,2(R8)      ; Indicate that we're done with node
                   00AD    31  0622           BRW     60$                         ; Skip the rest
                               0625      40$:
   00CF'CF    00C7'CF      28  0625           MOVC3   UETP$CLIG,UETP$CLIG+8,-     ; Get the full name...
              0CC4'CF          062C                   BUFFER
   63    0042'CF   06      28  062F           MOVC3   #NODE_LENGTH,SCSNODE,(R3);  ...
            83     5F 8F   90  0635           MOVB    #^A/_/,(R3)+                ; ...
      63    04 B8    06    28  0639           MOVC3   #NODE_LENGTH,@4(R8),(R3)    ; ...of the resource...
            83     5F 8F   90  063E           MOVB    #^A/_/,(R3)+                ; ...that the slave...
      63    04 B8    06    28  0642           MOVC3   #NODE_LENGTH,@4(R8),(R3)    ; ...supposedly just locked
         54   0CC4'CF      DE  0647           MOVAL   BUFFER,R4                   ; Fix up a descriptor...
   0CBC'CF    53     54    C3  064C           SUBL3   R4,R3,BUFFER_PTR            ; ...to the resource name
         50   0CBC'CF      DE  0652           MOVAL   BUFFER_PTR,R0
                               0657           $FAO_S  CTRSTR = DEBUG_REQ_LOCK_MSG,- ; Set up a program trace msg
                               0657                   OUTLEN = DEBUG_PTR,-
                               0657                   OUTBUF = DEBUG_FAO_BUF,-
                               0657                   P1     = R8,-
                               0657                   P2     = R0
                   1538    30  066E           BSBW    GIVE_DEBUG_MSG              ; Issue it, if appropriate
                               0671           $ENQ_S  LKMODE = #LCK$K_EXMODE,-    ; Is it a true lock?
                               0671                   LKSB   = QUAD_STATUS,-
                               0671                   FLAGS  = #LCK$M_NOQUEUE,-
                               0671                   RESNAM = BUFFER_PTR
         50   0000'8F      B1  068E           CMPW    #SS$_NOTQUEUED,R0           ; It will be...
                     3D    13  0693           BEQL    60$                         ; ..if we can't get it
                     50    DD  0695           PUSHL   R0
            1BC3'CF  01    FB  0697           CALLS   #1,STATUS_TO_TEXT           ; Get text for our result
                               069C           $FAO_S  CTRSTR = WRONG_ENQ,-        ; Form an explanatory message...
                               069C                   OUTLEN = BUFFER_PTR,-
                               069C                   OUTBUF = FAO_BUF,-
                               069C                   P1     = R8
                0EDE'CF    DF  06B1           PUSHAL  STATUS_PTR
                     01    DD  06B5           PUSHL   #1
           00741132 8F     DD  06B7           PUSHL   #UETP$_TEXT!STS$K_ERROR
                0CBC'CF    DF  06BD           PUSHAL  BUFFER_PTR
           000F0001 8F     DD  06C1           PUSHL   #^XF0001
           00741132 8F     DD  06C7           PUSHL   #UETP$_TEXT!STS$K_ERROR
            1DAD'CF  06    FB  06CD           CALLS   #6,ERROR_SIGNAL             ; ...and signal the error
                               06D2      60$:
                     87    B5  06D2           TSTW    (R7)+                       ; Point to the next possible channel
                     88    73  06D4           TSTD    (R8)+                       ; Point to the next possible name desc
                   FEE9    31  06D6           BRW     10$                         ; Loop to request the next lock
```

```
                               06D9           .SBTTL  TAKE_OUT_LOCK - Get a Lock at Master's Request
                               06D9    ;++
                               06D9    ; FUNCTIONAL DESCRIPTION:
                               06D9    ;       To test that locks are indeed cluster-wide the master process will
                               06D9    ;       request us to get a lock.  Report back the eventual status of that lock.
                               06D9    ;
                               06D9    ; IMPLICIT INPUTS:
                               06D9    ;       Name of a resource for us to lock, by way of message from master
                               06D9    ;               process.
                               06D9    ;
                               06D9    ; IMPLICIT OUTPUTS:
                               06D9    ;       NONE
                               06D9    ;
                               06D9    ; SIDE EFFECTS:
                               06D9    ;       Resource name is locked.
                               06D9    ;
                               06D9    ;--
                               06D9
                               06D9    TAKE_OUT_LOCK:
          59   0DBF'CF   DE    06D9            MOVAL   TAKELOCK_MSG,R9          ; Set up convenience registers...
          5A   0DC9'CF   DE    06DE            MOVAL   GOTLOCK_MSG,R10          ; ...
                    59   DD    06E3            PUSHL   R9                       ; Define the type of message we want
             16D0'CF  01   FB    06E5            CALLS   #1,SLAVE_READ            ; Get the master node's message
  0AA2'CF    02 A9    69   29    06EA            CMPC3   (R9),2(R9),MESSAGE_BUFFER ; What does the message say?
                    1C   13    06F1            BEQL    10$                      ; BR if it says ''TAKELOCK''
               00BB'CF   DF    06F3            PUSHAL  NULL                     ; Otherwise,...
               0094'CF   DF    06F7            PUSHAL  MASTER_NODE_DESC
                    59   DD    06FB            PUSHL   R9
             1B47'CF  03   FB    06FD            CALLS   #3,GARBLED_TRANS         ; ...signal the error
                               0702            $EXIT_S CODE = #UETP$_ABENDD!STS$K_ERROR!STS$M_INHIB_MSG
                               070F    10$:
             5B    53    D0    070F            MOVL    R3,R11                   ; Save ptr to resource name in msg
  00CF'CF    00C7'CF    28    0712            MOVC3   UETP$CLIG,UETP$CLIG+8,-  ; Set up...
             0CC4'CF          0719                    BUFFER
                    06   28    071C            MOVC3   #NODE_LENGTH,-           ; ...
          63   009C'CF         071E                    MASTER_NODE,(R3)
       83    5F 8F    90    0722            MOVB    #^A/_/,(R3)+             ; ...
          63    6B    06   28    0726            MOVC3   #NODE_LENGTH,(R11),(R3) ; ...the resource name...
       83    5F 8F    90    072A            MOVB    #^A/_/,(R3)+             ; ...
          63    6B    06   28    072E            MOVC3   #NODE_LENGTH,(R11),(R3) ; ...that we're supposed to lock
          54   0CC4'CF   DE    0732            MOVAL   BUFFER,R4                ; Set up a pointer...
  0CBC'CF    53    54    C3    0737            SUBL3   R4,R3,BUFFER_PTR         ; ...to that name
          50   0CBC'CF   DE    073D            MOVAL   BUFFER_PTR,R0
                               0742            $FAO_S  CTRSTR = DEBUG_TAK_LOCK_MSG,- ; Set up a program trace msg
                               0742                    OUTLEN = DEBUG_PTR,-
                               0742                    OUTBUF = DEBUG_FAO_BUF,-
                               0742                    P1     = R0
                  144F   30    0757            BSBW    GIVE_DEBUG_MSG           ; Issue it, if appropriate
                               075A            $ENQ_S  LKMODE = #LCK$K_EXMODE,- ; Try to lock the resource
                               075A                    LKSB   = QUAD_STATUS,-
                               075A                    FLAGS  = #LCK$M_NOQUEUE,-
                               075A                    RESNAM = BUFFER_PTR
     002C'CF     00'   B1    0777            CMPW    S^#SS$_NORMAL,QUAD_STATUS ; Did we ge the lock?
                    27   13    077C            BEQL    20$                      ; BR if so - we're OK
          7E   002C'CF   3C    077E            MOVZWL  QUAD_STATUS,-(SP)
     1BC3'CF    01   FB    0783            CALLS   #1,STATUS_TO_TEXT        ; Get text for our result
             0EDE'CF   DF    0788            PUSHAL  STATUS_PTR
```

```
                   01   DD  078C   1366          PUSHL   #1
        00741132   8F   DD  078E   1367          PUSHL   #UETP$_TEXT!STS$K_ERROR
             0545'CF    DF  0794   1368          PUSHAL  NO_LOCK_ENQ
                   01   DD  0798   1369          PUSHL   #1
        00741132   8F   DD  079A   1370          PUSHL   #UETP$_TEXT!STS$K_ERROR
                   06   DD  07A0   1371          PUSHL   #6
                 165B   31  07A2   1372          BRW     ERROR_EXIT              ; Signal error and exit
                            07A5   1373  20$:
          02 AA    6A   28  07A5   1374          MOVC3   (R10),2(R10),-            ; Set up msg telling master node...
              0AA2'CF       07A9   1375                  MESSAGE_BUFFER
     63   0042'CF  06   28  07AC   1376          MOVC3   #NODE_LENGTH,SCSNODE,(R3) ; ...that I got the lock
                   5A   DD  07B2   1377          PUSHL   R10                     ; Define the type of message we want
         1769'CF   01   FB  07B4   1378          CALLS   #1,SLAVE_WRITE          ; Tell master node the lock is OK
                        05  07B9   1379          RSB
```

```
                              07BA         .SBTTL  CHECK_DEADLOCK - See If Deadlock Detection Works
                              07BA  ;++
                              07BA  ; FUNCTIONAL DESCRIPTION:
                              07BA  ;      Using the locks taken out by CHECK_LOCKS, assign to each node a lock
                              07BA  ;      taken by another node.  This should result in a a chain of locks
                              07BA  ;      leading to a deadlock.  Check for a victim or timeout.  Ensure that
                              07BA  ;      deadlock detection was consistent throughout the cluster.  Use blocking
                              07BA  ;      ASTs to minimize the wait ot see if deadlock detection has occurred.
                              07BA  ;
                              07BA  ; IMPLICIT INPUTS:
                              07BA  ;      Set of locks taken during CHECK_LOCKS
                              07BA  ;
                              07BA  ; IMPLICIT OUTPUTS:
                              07BA  ;      NONE
                              07BA  ;
                              07BA  ; SIDE EFFECTS:
                              07BA  ;      NONE
                              07BA  ;
                              07BA  ;--
                              07BA
                              07BA  CHECK_DEADLOCK:
           007C'CF        D5  07BA         TSTL    DEADLOCK_WAIT              ; Is deadlock detection...
                2D        12  07BE         BNEQ    5$                         ; ...enabled for this node?  BR if so
      55   0042'CF        DE  07C0         MOVAL   SCSNODE,R5
                              07C5         $FAO_S  CTRSTR = DEADLOCK_OFF_MSG,- ; Warn if not
                              07C5                 OUTLEN = BUFFER_PTR,-
                              07C5                 OUTBUF = FAO_BUF,-
                              07C5                 P1     = #NODE_LENGTH,-
                              07C5                 P2     = R5
                              07DC         $PUTMSG_S MSGVEC = DEADLOCK_OFF_PTR
                              07ED  5$:
                56        D4  07ED         CLRL    R6                         ; This will index through nodes...
                              07EF                                            ; ...for the resource a slave is...
                              07EF                                            ; ...to lock during this step
                57        D4  07EF         CLRL    R7                         ; This will index through nodes...
                              07F1                                            ; ...for the slave that is to...
                              07F1                                            ; ...take out the lock
                5C        D4  07F1         CLRL    R12                        ; If non-zero, we have found...
                              07F3                                            ; ...some nodes for deadlock check
           0080'CF        D4  07F3         CLRL    DEADLOCK_COUNT             ; Counts deadlock participants who...
                              07F7                                            ; ...have not yet caused us a...
                              07F7                                            ; ...blocking AST
      59   0DBF'CF        DE  07F7         MOVAL   TAKELOCK_MSG,R9            ; Set up convenience registers...
      5A   0DD2'CF        DE  07FC         MOVAL   QUEUELOCK_MSG,R10          ; ...
  00    02 A9     69      2C  0801         MOVC5   (R9),2(R9),#0,-            ; Set up msg telling slaves...
           010D 8F            0806                 #TEXTB_SIZE,-              ; ...to take out a lock
           0AA2'CF            0809                 MESSAGE_BUFFER
 00CF'CF   00C7'CF        28  080C         MOVC3   UETP$CLIG,UETP$CLIG+8,-    ; Form a name...
           0CC4'CF            0813                 BUFFER
 63   0042'CF     06      28  0816         MOVC3   #NODE_LENGTH,SCSNODE,(R3)  ; ...for a lock that we'll hold...
63 00DD'DF 00D9'CF        28  081C         MOVC3   BLOCK,@BLOCK+4,(R3)        ; ...which will result in...
      54   0CC4'CF        DE  0824         MOVAL   BUFFER,R4                  ; ...a blocking AST...
 0CBC'CF   53     54      C3  0829         SUBL3   R4,R3,BUFFER_PTR           ; ...whenever a slave tries to get it
                              082F         $ENQ_S  LKMODE = #LCK$K_EXMODE,-  ; We'll use this lock...
                              082F                 LKSB   = QUAD_STATUS,-     ; ...and the blocking ASTs from it...
                              082F                 FLAGS  = #LCK$M_NOQUEUE,-
                              082F                 RESNAM = BUFFER_PTR,-      ; ...to count slaves who don't yet...
```

```
                                082F              BLKAST = 200$                 ; ...know if they are deadlock victims
             0030'CF     D0     084E         MOVL    QUAD_STATUS+4,-            ; Save lock id so we can requeue BLKAST
             0084'CF            0852                 DEADLOCK_LOCKID
          2A 002C'CF     E8     0855         BLBS    QUAD_STATUS,10$            ; BR if we're correctly set up
             002C'CF     DD     085A         PUSHL   QUAD_STATUS
         1BC3'CF    01   FB     085E         CALLS   #1,STATUS_TO_TEXT          ; Get text of error status
             0EDE'CF     DF     0863         PUSHAL  STATUS_PTR
                  01     DD     0867         PUSHL   #1
          00741132 8F    DD     0869         PUSHL   #UETP$_TEXT!STS$K_ERROR
             0583'CF     DF     086F         PUSHAL  NO_BLOCK_LOCK              ; It won't affect deadlock detection...
          000F0001 8F    DD     0873         PUSHL   #^XF0001
          00741132 8F    DD     0879         PUSHL   #UETP$_TEXT!STS$K_ERROR
         1DAD'CF    06   FB     087F         CALLS   #6,ERROR_SIGNAL            ; ...but it's worth letting users know
                                0884 10$:
           00AA'CF47     B5     0884         TSTW    NODE_CHANS[R7]             ; Have we another channel?
                                0889         BEQLW   100$                       ; BR if not - check deadlock
      54   02AA'CF47     7E     088E         MOVAQ   NODE_NAMES[R7],R4
                                0894         BBSW    #CLIG_V_DEADNODE,2(R4),90$ ; BR to next node if this one is dead
                                089C ;
                                089C ; Note that if we get here there exists at least one node such that we have
                                089C ; a DECnet channel assigned to it and that we know the node is not dead.  That
                                089C ; means that we need have no concern over an endless loop in picking a
                                089C ; resource name to lock, given that the resource name will be the name of
                                089C ; some node.
                                089C ;
                  5C     D6     089C         INCL    R12                        ; Indicate that a node was found
             0080'CF     D6     089E         INCL    DEADLOCK_COUNT             ; This node hasn't casued us an AST yet
                  56     D6     08A2         INCL    R6                         ; Init to choose the node name...
                                08A4                                            ; ...for next resource to lock
                                08A4 20$:
           00AA'CF46     B5     08A4         TSTW    NODE_CHANS[R6]             ; Have we reached the end of the list?
                  13     13     08A9         BEQL    30$                        ; BR if so - we'll wrap around
      54   02AA'CF46     7E     08AB         MOVAQ   NODE_NAMES[R6],R4
                  01     E1     08B1         BBC     #CLIG_V_DEADNODE,-         ; BR if this node will be available...
            0C 02 A4            08B3                 2(R4),40$                  ; ...to take a lock of its own
   E6 56   000000FF 8F   F2     08B6         AOBLSS  #MAX_NODES,R6,20$          ; Point to the next possible node
                                08BE 30$:
                  56     D4     08BE         CLRL    R6                         ; We've wrapped around in our chain
                  E2     11     08C0         BRB     20$                        ; Wrap around in our search
                                08C2 ;
                                08C2 ; We have a slave node ([R7]) available to take out a lock and a slave node
                                08C2 ; ([R6], possibly the same one in a one-node cluster or if there have been
                                08C2 ; errors) which should already have that lock.
                                08C2 ;
                                08C2 40$:
      54   02AA'CF46     7E     08C2         MOVAQ   NODE_NAMES[R6],R4
               50   69   3C     08C8         MOVZWL  (R9),R0                    ; Append node name to the message...
      50   0AA2'CF40     9E     08CB         MOVAB   MESSAGE_BUFFER[R0],R0      ; ...
     60    04 B4    06   28     08D1         MOVC3   #NODE_LENGTH,4(R4),(R0)    ; ...so slave knows resource to lock
      7E   00AA'CF47     3C     08D6         MOVZWL  NODE_CHANS[R7],-(SP)       ; Set up the channel...
           02AA'CF47     7F     08DC         PUSHAQ  NODE_NAMES[R7]             ; ...the node name...
                  59     DD     08E1         PUSHL   R9                         ; ...and our message name
         1922'CF    03   FB     08E3         CALLS   #3,MASTER_WRITE            ; Tell this node to get a lock
                                08E8         BLBCW   R0,80$                     ; Skip the rest if this node died
      7E   00AA'CF47     3C     08EE         MOVZWL  NODE_CHANS[R7],-(SP)       ; Set up the channel...
           02AA'CF47     7F     08F4         PUSHAQ  NODE_NAMES[R7]             ; ...the node name...
                  5A     DD     08F9         PUSHL   R10                        ; ...and our message name
          19B0'CF   03  FB  08FB           CALLS   #3,MASTER_READ              ; See if this node got the lock
                            0900           BLBCW   R0,80$                      ; Error in sending, skip the rest
  0CC4'CF   02 AA   6A  29  0906           CMPC3   (R10),2(R10),BUFFER         ; Did we get the reply we wanted?
                    0D  12  090D           BNEQ    50$                         ; BR if not
       54   02AA'CF47   7E  090F           MOVAQ   NODE_NAMES[R7],R4
     63    04 B4    64  29  0915           CMPC3   (R4),@4(R4),(R3)            ; Was reply from the node we wanted?
                    1D  13  091A           BEQL    60$                         ; BR if it was
                            091C      50$:
          0999'CF       DF  091C           PUSHAL  EXCLUDE_MSG                 ; Complain that we got back trash
        02AA'CF47       7F  0920           PUSHAQ  NODE_NAMES[R7]
                    5A  DD  0925           PUSHL   R10
         1B47'CF    03  FB  0927           CALLS   #3,GARBLED_TRANS
       54   02AA'CF47   7E  092C           MOVAQ   NODE_NAMES[R7],R4
          02 A4     02  A8  0932           BISW2   #CLIG_M_DEADNODE,2(R4)      ; Indicate that we're done with node
                  0131  31  0936           BRW     80$                         ; Skip the rest
                            0939      60$:
          0CD3'CF       D0  0939           MOVL    BUFFER+QUEUELOCK_LENGTH+- ; Get this node's dlock wait interval
                    53      093D                   NODE_LENGTH,R3
       54   02AA'CF47   7E  093E           MOVAQ   NODE_NAMES[R7],R4           ; Set up for possible message
     53     007C'CF     D1  0944           CMPL    DEADLOCK_WAIT,R3            ; Is deadlock checking consistent?
                    39  13  0949           BEQL    70$                         ; BR if it is
      55    0042'CF     DE  094B           MOVAL   SCSNODE,R5
                            0950           $FAO_S  CTRSTR = DEADLOCK_WAIT_MSG,- ; Complain if it isn't
                            0950                   OUTLEN = BUFFER_PTR,-
                            0950                   OUTBUF = FAO_BUF,-
                            0950                   P1     = R3,-
                            0950                   P2     = R4,-
                            0950                   P3     = DEADLOCK_WAIT,-
                            0950                   P4     = #NODE_LENGTH,-
                            0950                   P5     = R5
          0CBC'CF       DF  096F           PUSHAL  BUFFER_PTR
      000F0001  8F      DD  0973           PUSHL   #^XF0001
      00741132  8F      DD  0979           PUSHL   #UETP$_TEXT!STS$K_ERROR
         1DAD'CF    03  FB  097F           CALLS   #3,ERROR_SIGNAL
                            0984      70$:
                    53  D5  0984           TSTL    R3                          ; Is deadlock detection...
                    29  12  0986           BNEQ    75$                         ; ...enabled for this node?  BR if so
                            0988           $FAO_S  CTRSTR = DEADLOCK_OFF_MSG,- ; Warn if not
                            0988                   OUTLEN = BUFFER_PTR,-
                            0988                   OUTBUF = FAO_BUF,-
                            0988                   P1     = (R4),-
                            0988                   P2     = 4(R4)
                            09A0           $PUTMSG_S MSGVEC = DEADLOCK_OFF_PTR
                            09B1      75$:
   00CF'CF     00C7'CF  28  09B1           MOVC3   UETP$CLIG,UETP$CLIG+8,-   ; Get the full name...
               0CC4'CF      09B8                   BUFFER
   63   0042'CF     06  28  09B8           MOVC3   #NODE_LENGTH,SCSNODE,(R3) ; ...
          83   5F   8F  90  09C1           MOVB    #^A/_7,(R3)+              ; ...
     58     02AA'CF46   7E  09C5           MOVAQ   NODE_NAMES[R6],R8         ; ...
     63    04 B8    06  28  09CB           MOVC3   #NODE_LENGTH,@4(R8),(R3)  ; ...of the resource...
          83   5F   8F  90  09D0           MOVB    #^A/_7,(R3)+              ; ...that the slave...
     63    04 B8    06  28  09D4           MOVC3   #NODE_LENGTH,@4(R8),(R3)  ; ...supposedly just locked
       54   0CC4'CF     DE  09D9           MOVAL   BUFFER,R4                 ; Fix up a descriptor...
   0CBC'CF    53    54  C3  09DE           SUBL3   R4,R3,BUFFER_PTR          ; ...to the resource name
       50   0CBC'CF     DE  09E4           MOVAL   BUFFER_PTR,R0
     54     02AA'CF47   7E  09E9           MOVAQ   NODE_NAMES[R7],R4         ; Get address of node name desc
                            09EF           $FAO_S  CTRSTR = DEBUG_REQ_LOCK_MSG,- ; Set up a program trace msg
```

```
                           09EF                          OUTLEN = DEBUG_PTR,-
                           09EF                          OUTBUF = DEBUG_FAO_BUF,-
                           09EF                          P1     = R4,-
                           09EF                          P2     = R0
           11A0      30    0A06              BSBW        GIVE_DEBUG_MSG           ; Issue it, if appropriate
                           0A09              $ENQ_S      LKMODE = #LCK$K_EXMODE,- ; Is it a true lock?
                           0A09                          LKSB   = QUAD_STATUS,-
                           0A09                          FLAGS  = #LCK$M_NOQUEUE,-
                           0A09                          RESNAM = BUFFER_PTR
  50   0000'8F       B1    0A26              CMPW        #SS$_NOTQUEUED,R0        ; It will be...
             4E      13    0A2B              BEQL        90$                      ; ..if we can't get it
             50      DD    0A2D              PUSHL       R0
1BC3'CF      01      FB    0A2F              CALLS       #1,STATUS_TO_TEXT        ; Get text for our result
                           0A34              $FAO_S      CTRSTR = WRONG_ENQ,-     ; Form an explanatory message...
                           0A34                          OUTLEN = BUFFER_PTR,-
                           0A34                          OUTBUF = FAO_BUF,-
                           0A34                          P1     = R4
        0EDE'CF      DF    0A49              PUSHAL      STATUS_PTR
             01      DD    0A4D              PUSHL       #1
   00741132 8F       DD    0A4F              PUSHL       #UETP$_TEXT!STS$K_ERROR
        0CBC'CF      DF    0A55              PUSHAL      BUFFER_PTR
   000F0001 8F       DD    0A59              PUSHL       #^XF0001
   00741132 8F       DD    0A5F              PUSHL       #UETP$_TEXT!STS$K_ERROR
1DAD'CF      06      FB    0A65              CALLS       #6,ERROR_SIGNAL          ; ...and signal the error
                           0A6A         80$:
                           0A6A
                           0A6A              $PUTMSG_S MSGVEC = -                 ; Warn that deadlock detection...
                           0A6A                               NO_DLOCK_SETUP_PTR  ; ...testing may fail
                           0A7B         90$:
             57      D6    0A7B              INCL        R7                       ; Point to the next possible node
           FE04      31    0A7D              BRW         10$                      ; Loop to request the next lock
                           0A80         ; Deadlock detection checking continues on next page
```

```
                                                        I 9
UETCLIG00                    VAX/VMS UETP Cluster Integration Test    16-SEP-1984 00:19:09  VAX/VMS Macro V04-00         Page  38
V04-000                      CHECK_DEADLOCK - See If Deadlock Detecti  6-SEP-1984 10:00:47  [UETPSY.SRC]UETCLIG00.MAR;1      (15)

                                0A80     ;
                                0A80     ; Each surviving node has been told to take out a lock on a resource held
                                0A80     ; by some other node, a situation that should result in deadlock.  Wait
                                0A80     ; long enough for deadlock to have been detected and a message sent to us
                                0A80     ; to that effect.  See if deadlock was properly detected.
                                0A80     ;
                                0A80     100$:
                     5C   D5    0A80             TSTL    R12                      ; Did we find any nodes for deadlock?
                                0A82             BEQLW   140$                     ; BR if not
            00000078 8F   C1    0A87             ADDL3   #2*QIO_TIMEOUT,-         ; Compute a time to wait...
            50  007C'CF         0A8D                     DEADLOCK_WAIT,R0         ; ...to hear about a victim process
  00    50  FF676980 8F   7A    0A91             EMUL    #-10000000,R0,#0,-       ; Convert seconds to delta time
                0088'CF         0A99                     DEADLOCK_MSG_TIME
                                0A9C             $SCHDWK_S DAYTIM = -             ; Wait for some process to be chosen
                                0A9C                             DEADLOCK_MSG_TIME
                                0AAD             $SETAST_S ENBFLG = #0            ; BLKAST during next code would be bad
                0080'CF   D5    0AB6             TSTL    DEADLOCK_COUNT           ; Any slaves who don't yet know if...
                     17   13    0ABA             BEQL    105$                     ; ...they're deadlock victim? BR if not
                0080'CF   CE    0ABC             MNEGL   DEADLOCK_COUNT,-         ; Indicate that we can $WAKE from $HIBER
                0080'CF         0AC0                     DEADLOCK_COUNT
                                0AC3             $SETAST_S ENBFLG = #1            ; End of non-interruptible code
                                0ACC             $HIBER_S
                                0AD3
                                0AD3     105$:
                                0AD3             $SETAST_S ENBFLG = #1            ; DEADLOCK_COUNT is consistent again
                                0ADC             $CANWAK_S                        ; We may have aWAKEned early from $HIBER
            57  00AA'CF   3E    0AE7             MOVAW   NODE_CHANS,R7            ; Used to loop through DECnet channels
            58  02AA'CF   7E    0AEC             MOVAQ   NODE_NAMES,R8            ; Used to loop through node name descs
            5A  0DDD'CF   DE    0AF1             MOVAL   DEADLOCK_MSG,R10         ; Set up convenience register
                                0AF6     110$:
                     67   B5    0AF6             TSTW    (R7)                     ; Have we another channel?
                     27   13    0AF8             BEQL    130$                     ; BR if not - check results of our poll
                     01   E0    0AFA             BBS     #CLIG_V_DEADNODE,-       ; Skip trying to read from this node...
               1C 02 A8         0AFC                     2(R8),120$               ; ...if we already know it's broken
                7E   67   3C    0AFF             MOVZWL  (R7),-(SP)               ; Set up the channel...
                     58   DD    0B02             PUSHL   R8                       ; ...the node name...
                     5A   DD    0B04             PUSHL   R10                      ; ...and our message name
           19B0'CF   03   FB    0B06             CALLS   #3,MASTER_READ           ; See if this node was deadlock victim
                0D   50   E9    0B0B             BLBC    R0,120$                  ; Skip the rest if DECnet error
  0CC4'CF    02 AA   6A   29    0B0E             CMPC3   (R10),2(R10),BUFFER      ; Was this node a victim?
                     04   12    0B15             BNEQ    120$                     ; BR if not
                0078'CF   D6    0B17             INCL    DEADLOCK_VICTIMS         ; Count it if it was
                                0B1B     120$:
                     87   B5    0B1B             TSTW    (R7)+                    ; Point to the next possible channel
                     88   73    0B1D             TSTD    (R8)+                    ; Point ot the next possible name desc
                     D5   11    0B1F             BRB     110$                     ; Loop to poll the next one
                                0B21
                                0B21     130$:
           0078'CF   01   D1    0B21             CMPL    #1,DEADLOCK_VICTIMS      ; Have we exactly one deadlock victim?
                     2C   13    0B26             BEQL    140$                     ; BR if so - all is OK
                                0B28             $FAO_S  CTRSTR = VICTIMS_MSG,-   ; Make a noise if not
                                0B28                     OUTLEN = BUFFER_PTR,-
                                0B28                     OUTBUF = FAO_BUF,-
                                0B28                     P1     = DEADLOCK_VICTIMS
                0CBC'CF   DF    0B3F             PUSHAL  BUFFER_PTR
            000F0001 8F   DD    0B43             PUSHL   #^XF0001
            00741132 8F   DD    0B49             PUSHL   #UETP$_TEXT!STS$K_ERROR
```

```
                1DAD'CF   03  FB  0B4F  1642        CALLS   #3,ERROR_SIGNAL
                                  0B54  1643 140$:
                              05  0B54  1644        RSB
```

```
                           0B55  ;
                           0B55  ; AST routine for blocking AST from a slave process when that slave has
                           0B55  ; discovered whether or not it's a deadlock victim.  We'll keep track of
                           0B55  ; the number of slaves who don't yet know and limit the time the master
                           0B55  ; process $HIBERnates while waiting to be told.
                           0B55  ;
                           0B55  200$:
                     0000  0B55          .WORD   ^M<>
                           0B57
 12 0080'CF    1F    E1    0B57          BBC     #31,DEADLOCK_COUNT,210$ ; BR if master is not going to $HIBER
       0080'CF       D6    0B5D          INCL    DEADLOCK_COUNT          ; We're $HIBERnating. Count down...
            10       12    0B61          BNEQ    220$                    ; ...and BR if tally is not final
                           0B63          $WAKE_S                         ; All slaves have reported back
                     04    0B6E          RET
                           0B6F  210$:
       0080'CF       D7    0B6F          DECL    DEADLOCK_COUNT          ; Slave reported back quickly
                           0B73  220$:                                   ; We don't know if we have final...
       0084'CF       D0    0B73          MOVL    DEADLOCK_LOCKID,-       ; ...yet, so we must re-enable...
       0030'CF             0B77                  QUAD_STATUS+4           ; ...BLKAST for other slaves
                           0B7A          $ENQW_S EFN    = #SS_SYNCH_EFN,- ; Set up BLKAST for another slave
                           0B7A                  LKMODE = #LCK$K_EXMODE,-
                           0B7A                  LKSB   = QUAD_STATUS,-
                           0B7A                  FLAGS  = #LCK$M_CONVERT,-
                           0B7A                  BLKAST = 200$
                     04    0B96          RET
```

```
                                  0B97          .SBTTL  GET_DEADLOCK - Participate in a Cluster-Wide Deadlock
                                  0B97  ;++
                                  0B97  ; FUNCTIONAL DESCRIPTION:
                                  0B97  ;       See if cluster-wide deadlock detection works.  Take out another lock
                                  0B97  ;       at the master's request.  This one should ultimately result in a
                                  0B97  ;       deadlock, though.
                                  0B97  ;
                                  0B97  ; IMPLICIT INPUTS:
                                  0B97  ;       Name of a resource for us to lock, by way of message from master
                                  0B97  ;               process.
                                  0B97  ;
                                  0B97  ; IMPLICIT OUTPUTS:
                                  0B97  ;       NONE
                                  0B97  ;
                                  0B97  ; SIDE EFFECTS:
                                  0B97  ;       Resource name is locked.
                                  0B97  ;       Deadlock or timeout.
                                  0B97  ;
                                  0B97  ;--
                                  0B97
                                  0B97  GET_DEADLOCK:
              59   0DBF'CF    DE  0B97          MOVAL   TAKELOCK_MSG,R9              ; Set up convenience registers...
              5A   0DD2'CF    DE  0B9C          MOVAL   QUEUELOCK_MSG,R10            ; ...
                        59    DD  0BA1          PUSHL   R9                           ; Define the type of message we want
            16D0'CF     01    FB  0BA3          CALLS   #1,SLAVE_READ                ; Get the master node's message
   0AA2'CF   02 A9      69    29  0BA8          CMPC3   (R9),2(R9),MESSAGE_BUFFER    ; What does the message say?
                        1C    13  0BAF          BEQL    10$                          ; BR if it says "TAKELOCK"
               00BB'CF        DF  0BB1          PUSHAL  NULL                         ; Otherwise,...
               0094'CF        DF  0BB5          PUSHAL  MASTER_NODE_DESC
                        59    DD  0BB9          PUSHL   R9
            1B47'CF     03    FB  0BBB          CALLS   #3,GARBLED_TRANS             ; ...signal the error
                                  0BC0          $EXIT_S CODE = #UETP$_ABENDD!STS$K_ERROR!STS$M_INHIB_MSG
                                  0BCD  10$:
              5B        53    D0  0BCD          MOVL    R3,R11                       ; Save ptr to resource name in msg
   00CF'CF     00C7'CF        28  0BD0          MOVC3   UETP$CLIG,UETP$CLIG+8,-      ; Set up...
               0CC4'CF            0BD7                  BUFFER
                        06    28  0BDA          MOVC3   #NODE_LENGTH,-               ; ...
              63   009C'CF        0BDC                  MASTER_NODE,(R3)
                83   5F 8F    90  0BE0          MOVB    #^A/ /,(R3)+                 ; ...
              63   6B   06    28  0BE4          MOVC3   #NODE_LENGTH,(R11),(R3)      ; ...the resource name...
                83   5F 8F    90  0BE8          MOVB    #^A/ /,(R3)+                 ; ...
              63   6B   06    28  0BEC          MOVC3   #NODE_LENGTH,(R11),(R3)      ; ...that we're supposed to lock
              54   0CC4'CF    DE  0BF0          MOVAL   BUFFER,R4                    ; Set up a pointer...
   0CBC'CF     53       54    C3  0BF5          SUBL3   R4,R3,BUFFER_PTR             ; ...to that name
              50   0CBC'CF    DE  0BFB          MOVAL   BUFFER_PTR,R0
                                  0C00          $FAO_S  CTRSTR = DEBUG_TAK_LOCK_MSG,- ; Set up a program trace msg
                                  0C00                  OUTLEN = DEBUG_PTR,-
                                  0C00                  OUTBUF = DEBUG_FAO_BUF,-
                                  0C00                  P1     = R0
                  0F91        30  0C15          BSBW    GIVE_DEBUG_MSG               ; Issue it, if appropriate
                                  0C18          $SETAST_S ENBFLG = #0                ; Synch lock AST with DECnet writes
                                  0C21          $ENQ_S  LKMODE = #LCK$K_EXMODE,-     ; Try to lock the resource
                                  0C21                  LKSB   = QUAD_STATUS,-
                                  0C21                  RESNAM = BUFFER_PTR,-
                                  0C21                  ASTADR = 100$
              50        00'   B1  0C42          CMPW    S^#SS$_NORMAL,R0             ; Are we queued for the lock?
                        28    13  0C45          BEQL    20$                          ; BR if so - we're OK
```

```
                     50   DD  0C47           PUSHL    R0
          1BC3'CF    01   FB  0C49           CALLS    #1,STATUS_TO_TEXT          ; Get text for our result
             0EDE'CF      DF  0C4E           PUSHAL   STATUS_PTR
                     01   DD  0C52           PUSHL    #1
           00741132 8F    DD  0C54           PUSHL    #UETP$_TEXT!STS$K_ERROR
              06F9'CF     DF  0C5A           PUSHAL   DLOCK_ENQ
           000F0001 8F    DD  0C5E           PUSHL    #^XF0001
           00741132 8F    DD  0C64           PUSHL    #UETP$_TEXT!STS$K_ERROR
          1DAD'CF    06   FB  0C6A           CALLS    #6,ERROR_SIGNAL             ; Don't exit - we may be holding a...
                              0C6F                                               ; ...lock needed for deadlock
                              0C6F     20$:
             02 AA   6A   28  0C6F           MOVC3    (R10),2(R10),-              ; Set up msg telling master node...
             0AA2'CF          0C73                    MESSAGE_BUFFER
      63    0042'CF  06   28  0C76           MOVC3    #NODE_LENGTH,SCSNODE,(R3)  ; ...that I'm queued for the lock
          63   007C'CF    D0  0C7C           MOVL     DEADLOCK_WAIT,(R3)         ; Include deadlock checkirg interval
                     5A   DD  0C81           PUSHL    R10                        ; Define the type of message we want
          1769'CF    01   FB  0C83           CALLS    #1,SLAVE_WRITE             ; Tell master node that we're OK
                              0C88           $SETAST_S ENBFLG = #1               ; Synch lock AST with DECnet writes
           00000078 8F    C1  0C91           ADDL3    #2*QIO_TIMEOUT,-           ; Compute a time to wait...
          50   007C'CF        0C97                    DEADLOCK_WAIT,R0           ; ...to see if we got the lock
  00    50   FF676980 8F  7A  0C9B           EMUL     #-10000000,R0,#0,-         ; Convert seconds to delta time
             0088'CF          0CA3                    DEADLOCK_MSG_TIME
                              0CA6           $SETIMR_S EFN  = #SS_SYNCH_EFN,-  ; Wait for deadlock resolution
                              0CA6                    DAYTIM = DEADLOCK_MSG_TIME,-
                              0CA6                    ASTADR = 200$
                              0CB9           $HIBER_S
                              0CC0           $CANTIM_S                           ; Deadlock resolved or timer went off
      00CF'CF    00C7'CF  28  0CC9           MOVC3    UETP$CLIG,UETP$CLIG+8,-   ; Set up...
             0CC4'CF          0CD0                    BUFFER
                     06   28  0CD3           MOVC3    #NODE_LENGTH,-             ; ...the resource name...
          63   009C'CF        0CD5                    MASTER_NODE,(R3)
  63   00DD'DF   00D9'CF  28  0CD9           MOVC3    BLOCK,@BLOCK+4,(R3)        ; ...that the master has locked...
          54   0CC4'CF    DE  0CE1           MOVAL    BUFFER,R4                  ; ...in order to get blocking ASTs
      0CBC'CF    53  54   C3  0CE6           SUBL3    R4,R3,BUFFER_PTR
                              0CEC           $ENQ_S   LKMODE = #LCK$K_EXMODE,-  ; Try to lock the resource
                              0CEC                    LKSB   = QUAD_STATUS,-
                              0CEC                    RESNAM = BUFFER_PTR
             50      00'  B1  0D09           CMPW     S^#SS$_NORMAL,R0           ; Are we queued for the lock?
                     28   13  0D0C           BEQL     30$                        ; BR if so - we're OK
                     50   DD  0D0E           PUSHL    R0
          1BC3'CF    01   FB  0D10           CALLS    #1,STATUS_TO_TEXT          ; Get text for our result
             0EDE'CF      DF  0D15           PUSHAL   STATUS_PTR
                     01   DD  0D19           PUSHL    #1
           00741132 8F    DD  0D1B           PUSHL    #UETP$_TEXT!STS$K_ERROR
              0735'CF     DF  0D21           PUSHAL   NO_SLAVE_BLOCK
           000F0001 8F    DD  0D25           PUSHL    #^XF0001
           00741132 8F    DD  0D2B           PUSHL    #UETP$_TEXT!STS$K_ERROR
          1DAD'CF    06   FB  0D31           CALLS    #6,ERROR_SIGNAL             ; Don't exit - we may be holding a...
                              0D36                                               ; ...lock needed for deadlock
                              0D36     30$:
                          05  0D36           RSB
```

```
                            0D37     ;
                            0D37     ; AST routine for when deadlock is detected or lock request is otherwise
                            0D37     ; resolved.  If we timed out and already dequeued our locks, either deadlock
                            0D37     ; was not detected or other systems have been slow to dequ   [illegible]heir locks.
                            0D37     ; If we're the victim, everything is fine.  If we get our lock, some other
                            0D37     ; system must be the victim and everything is still fine.  In any case,
                            0D37     ; dequeue all locks.
                            0D37     ;
                            0D37     100$:
                      063C  0D37             .WORD   ^M<R2,R3,R4,R5,R9,R10>
                            0D39
        5A   0DDD'CF   DE   0D39             MOVAL   DEADLOCK_MSG,R10        ; Assume we're deadlock victim
        59   00BF'CF   7E   0D3E             MOVAQ   BLANK_LINE,R9
   002C'CF   0000'8F   B1   0D43             CMPW    #SS$_DEADLOCK,QUAD_STATUS ; But are we?
                  0A   13   0D4A             BEQL    110$                    ; BR if we are
        5A   0DD2'CF   DE   0D4C             MOVAL   QUEUELOCK_MSG,R10       ; Anything else is of no importance
        59   0B54'CF   7E   0D51             MOVAQ   NOT_MSG,R9
                            0D56     110$:
        50   0042'CF   DE   0D56             MOVAL   SCSNODE,R0
                            0D5B             $FAO_S  CTRSTR = DEBUG_DLOCK_VICTIM_MSG,- ; Set up a program trace msg
                            0D5B                     OUTLEN = DEBUG_PTR,-
                            0D5B                     OUTBUF = DEBUG_FAO_BUF,-
                            0D5B                     P1     = #NODE_LENGTH,-
                            0D5B                     P2     = R0,-
                            0D5B                     P3     = R9
                0E32   30   0D74             BSBW    GIVE_DEBUG_MSG          ; Issue it, if appropriate
 0AA2'CF     02 AA  6A   28  0D77             MOVC3   (R10),2(R10),MESSAGE_BUFFER ; Set up the message
                  5A   DD   0D7E             PUSHL   R10                     ; Send our status...
        1769'CF   01   FB   0D80             CALLS   #1,SLAVE_WRITE          ; ...to the master node
                            0D85             $DEQ_S  FLAGS = #LCK$M_DEQALL   ; Allow other nodes to get locks
                            0D94             $WAKE_S                         ; Allow the test to get going again
                        04  0D9F             RET
                            0DA0
                            0DA0
                            0DA0
                            0DA0
                            0DA0     ;
                            0DA0     ; The timer used to allow deadlock detection to occur has gone off.
                            0DA0     ; If we're not the victim or deadlock was not detected, releasing locks allows
                            0DA0     ; the AST from the $ENQ to be delivered.  We'll send a message to the
                            0DA0     ; master process from that AST routine.
                            0DA0     ;
                            0DA0     200$:
                      0000  0DA0             .WORD   ^M<>
                            0DA2
                            0DA2             $DEQ_S  FLAGS = #LCK$M_DEQALL   ; Allow other nodes to get locks
                        04  0DB1             RET
```

```
                         0DB2            .SBTTL  FILE_ACCESS - See If We Can Get to Cluster Files
                         0DB2      ;++
                         0DB2      ; FUNCTIONAL DESCRIPTION:
                         0DB2      ;         For each node in the cluster (NOT necessarily VMS node), create a
                         0DB2      ;         file on some disk local to that node.  The file will be in the
                         0DB2      ;         [SYSTEST] directory, which may or may not be in a rooted directory
                         0DB2      ;         (same algorithm as the UETP disk device test).  Warn if for some
                         0DB2      ;         reason we could not create the file.  Write, read, extend, share
                         0DB2      ;         access with a friend, and delete the file.
                         0DB2      ;
                         0DB2      ; IMPLICIT INPUTS:
                         0DB2      ;         The list of cluster nodes and devices from UETP$CLSIODB
                         0DB2      ;
                         0DB2      ; IMPLICIT OUTPUTS:
                         0DB2      ;         NONE
                         0DB2      ;
                         0DB2      ; SIDE EFFECTS:
                         0DB2      ;         Temporary file on various cluster accessible disks.  The file spec
                         0DB2      ;                 will look like:  test-node$ddcu:UETP$CLIG_master-node.TEST;1.
                         0DB2      ;
                         0DB2      ;--
                         0DB2
                         0DB2      ; R6 through R10 have specific purposes by this upper level routine.  They
                         0DB2      ; may be updated by some of the subroutines, but not trashed.
                         0DB2      FILE_ACCESS:
   56   00A2'CF    D0    0DB2              MOVL    CLSPTR,R6                    ; Point to SID records
                         0DB7      10$:
11 A6   0099'CF    D1    0DB7              CMPL    VMS,UID$SID$T_SWTYPE(R6) ; Is this a VAX/VMS node?
                         0DBD              BNEQW   20$                          ; BR if it is not - fewer tests
                         0DC2              $SETSFM_S ENBFLG = #0                ; Turn off SS errors
        32 A6      9F    0DCB              PUSHAB  UID$SID$T_NODENAME+1(R6)     ; Fix up a temp string descriptor...
   7E   31 A6      9A    0DCE              MOVZBL  UID$SID$T_NODENAME(R6),-(SP) ; ...for the node name...
   52      5E      D0    0DD2              MOVL    SP,R2                        ; ...and a pointer to it
                         0DD5              $GETSYIW_S EFN   = #SS_SYNCH_EFN,- ; ...while checking to see...
                         0DD5                      IOSB     = QUAD_STATUS,-   ; ...if this node is in our cluster
                         0DD5                      ITMLST   = OTHERNODE_ITMLST,-
                         0DD5                      NODENAME = (R2)
   5E      08      C0    0DEC              ADDL2   #8,SP                        ; Pop temp string descriptor from stack
   52      50      D0    0DEF              MOVL    R0,R2                        ; Preserve the return status...
                         0DF2              $SETSFM_S ENBFLG = #1                ; ...while resuming SS error checking
        21 52      E9    0DFB              BLBC    R2,30$                       ; BR if it is not a member
   1C 002C'CF      E9    0DFE              BLBC    QUAD_STATUS,30$              ; BR if it is not
   17 0090'CF      E9    0E03              BLBC    CLUSTER_MEMBER,30$           ; BR if it is not
                         0E08      20$:
   55   07 A6      D0    0E08              MOVL    UID$SID$L_PBFL(R6),R5        ; Have we any path to the node?
           11      13    0E0C              BEQL    30$                          ; BR if not
           03      B1    0E0E              CMPW    #PB$C_OPEN,-                 ; Is the path to this node open?
        07 A5            0E10                      UID$PATH$W_STATE(R5)
           0B      12    0E12              BNEQ    30$                          ; BR if not
   02      01      EF    0E14              EXTZV   #PB$V_STATE,#PB$S_STATE,- ; Is the path...
   54   0D A5            0E17                      UID$PATH$B_RSTATE(R5),R4
   54      02      91    0E1A              CMPB    #PB$C_ENAB,R4                ; ...to this node enabled?
           32      13    0E1D              BEQL    40$                          ; BR if it is
   5A   31 A6      9A    0E1F      30$:    MOVZBL  UID$SID$T_NODENAME(R6),R10   ; Get the length of the node name...
   59   32 A6      9E    0E23              MOVAB   UID$SID$T_NODENAME+1(R6),R9 ; ...and its address
                         0E27              $FAO_S  CTRSTR = MEMB_PATH,-     ; Complain that we can't...
                         0E27                      OUTLEN = BUFFER_PTR,-    ; ...test this node...
```

```
                           0E27                     OUTBUF = FAO_BUF,-       ; ...for remote file access
                           0E27                     P1     = R10,-
                           0E27                     P2     = R9
                           0E3E             $PUTMSG_S MSGVEC = MEMB_PATH_PTR
            78     11      0E4F             BRB     80$                      ; Loop for the next node
                           0E51      40$:
   57    41 A6     D0      0E51             MOVL    UIDSID$L_DDB(R6),R7      ; Get first possible DDB attached to SID
            09     13      0E55             BEQL    55$                      ; Don't process it if there are no DDBs
   58    07 A7     D0      0E57             MOVL    UIDDDB$L_UCB(R7),R8      ; Get the first UCB attached to DDB
                           0E5B      50$:
            78     10      0E5B             BSBB    100$                     ; Set up a FAB for a likely file
         32 50     E8      0E5D             BLBS    R0,60$                   ; BR if we have a candidate
                           0E60      55$:
   5A    31 A6     9A      0E60             MOVZBL  UIDSID$T_NODENAME(R6),R10  ; Get the length of the node name...
   59    32 A6     9E      0E64             MOVAB   UIDSID$T_NODENAME+1(R6),R9 ; ...and its address
                           0E68             $FAO_S  CTRSTR = NO_FILE_NODE,- ; Complain that we can't...
                           0E68                     OUTLEN = BUFFER_PTR,-    ; ...test this node...
                           0E68                     OUTBUF = FAO_BUF,-       ; ...for remote file access
                           0E68                     P1     = R10,-
                           0E68                     P2     = R9
                           0E7F             $PUTMSG_S MSGVEC = NO_FILE_NODE_PTR
            37     11      0E90             BRB     80$                      ; Loop to the next node
                           0E92      60$:
          0103     30      0E92             BSBW    200$                     ; See if we can create a file
         C3 50     E9      0E95             BLBC    R0,50$                   ; Get the next candidate if we can't
          0186     30      0E98             BSBW    300$                     ; Write and read a block of the file
         0D 50     E9      0E9B             BLBC    R0,70$                   ; Get rid of the file if we've an error
          01FE     30      0E9E             BSBW    400$                     ; Choose a slave to share access to file
         07 50     E9      0EA1             BLBC    R0,70$                   ; We're done with file if no sharing
            51     DD      0EA4             PUSHL   R1                       ; Value from 400$ routine is in R1
 1106'CF    01     FB      0EA6             CALLS   #1,500$                  ; Share access with a slave
                           0EAB      70$:
                           0EAB             $CLOSE  FAB = RF_FAB,-           ; We're done with this file...
                           0EAB                     ERR = RMS_ERROR
                           0EBA             $ERASE  FAB = RF_FAB,-           ; ...so get rid of it
                           0EBA                     ERR = RMS_ERROR
                           0EC9      80$:
      56    66     D0      0EC9             MOVL    UIDSID$A_FLINK(R6),R6    ; Point to the next possible SID record
                           0ECC             BNEQW   10$                      ; Loop for another node if there is one
          03B3     30      0ED1             BSBW    600$                     ; Tell all slaves to end file access
            05             0ED4             RSB
```

```
                                 0ED5  100$:                                       ; Set up a FAB for a likely file
                    58     D5    0ED5         TSTL    R8                           ; Have we run out of UCBs on this DDB?
                    10     13    0ED7         BEQL    110$                         ; BR if we have
                    00'    91    0ED9         CMPB    S^#DC$_DISK,-                ; Is this UCB for a disk?
                 09 A8           0EDB                 UIDUCB$B_DEVCLASS(R8)
                    0A     12    0EDD         BNEQ    110$                         ; BR if not
                    00'    E0    0EDF         BBS     S^#DEV$V_CLU,-               ; BR if the disk is cluster available
              15 0F A8           0EE1                 UIDUCB$L_DEVCHAR2(R8),130$
               58     68   D0    0EE4         MOVL    UIDUCB$A_FLINK(R8),R8        ; It's not,...
                      EC   11    0EE7         BRB     100$                         ; ...so try the next disk
                                 0EE9  110$:
               57     67   D0    0EE9         MOVL    UIDDDB$A_FLINK(R7),R7        ; Get next DDB - no shared disk UCB
                      57   D5    0EEC         TSTL    R7                           ; Have we run out of DDBs on this node?
                      03   12    0EEE         BNEQ    120$                         ; BR if not
                      50   D4    0EF0         CLRL    R0                           ; Indicate a problem if we have...
                           05    0EF2         RSB                                  ; ...and return with that error
                                 0EF3  120$:
          58     07 A7     D0    0EF3         MOVL    UIDDDB$L_UCB(R7),R8          ; Get the first UCB for this DDB
                      DC   11    0EF7         BRB     100$                         ; Check to see if it's OK
                                 0EF9  130$:
          50     31 A6     9B    0EF9         MOVZBW  UIDSID$T_NODENAME(R6),R0 ; Get the length of the node name
    1657'CF  50     02     81    0EFD         ADDB3   #2,R0,RF_FAB+FAB$B_FNS   ; Keep running count of it + overhead
          32 A6     50     28    0F03         MOVC3   R0,UIDSID$T_NODENAME+1(R6),- ; Move the nodename into filespec
            171F'CF              0F07                 RF_FILESPEC
               83   24     90    0F0A         MOVB    #^A/$/,(R3)+                 ; Append delimiter (overhead)
          50     08 A7     9B    0F0D         MOVZBW  UIDDDB$T_NAME(R7),R0         ; Get the length of the device name
         1657'CF    50     80    0F11         ADDB2   R0,RF_FAB+FAB$B_FNS          ; Keep a running count of spec length
      63     0C A7  50     28    0F16         MOVC3   R0,UIDDDB$T_NAME+1(R7),(R3) ; Concatenate the device name
     0CBC'CF        05     3C    0F1B         MOVZWL  #UNIT_LENGTH,BUFFER_PTR ; We have to get...
                    02     DD    0F20         PUSHL   #2                           ; ...
                    01     DD    0F22         PUSHL   #1                           ; ...
            0CBC'CF        7F    0F24         PUSHAQ  BUFFER_PTR                   ; ...
                 07 A8     3F    0F28         PUSHAW  UIDUCB$W_NUMBER(R8)          ; ...the device unit number...
  00000000'GF       04     FB    0F2B         CALLS   #4,G^OTS$CVT_L_TI            ; ...converted to text
 0CC4'CF     05     20     3B    0F32         SKPC    #^A/ /,#UNIT_LENGTH,BUFFER ; Strip leading blanks
        1657'CF     50     80    0F38         ADDB2   R0,RF_FAB+FAB$B_FNS          ; Keep a running count of spec length
          63   61   50     28    0F3D         MOVC3   R0,(R1),(R3)                 ; Concatenate the unit number
               83   3A     90    0F41         MOVB    #^A/:/,(R3)+                 ; Append delimiter (overhead)
   1657'CF   00C7'CF       80    0F44         ADDB2   UETP$CLIG,RF_FAB+FAB$B_FNS ; Keep the running count
63 00CF'CF   00C7'CF       28    0F4B         MOVC3   UETP$CLIG,UETP$CLIG+8,(R3) ; Concatenate part of filename
                 06   20   3A    0F53         LOCC    #^A/ /,#NODE_LENGTH,-        ; Strip trailing blanks...
             0042'CF             0F56                 SCSNODE                      ; ...from the master node name
        50   06     50     C3    0F59         SUBL3   R0,#NODE_LENGTH,R0           ; Get its true length
        1657'CF     50     80    0F5D         ADDB2   R0,RF_FAB+FAB$B_FNS          ; Keep a running count of spec length
   63   0042'CF     50     28    0F62         MOVC3   R0,SCSNODE,(R3)              ; Concatenate rest of the filename
   1657'CF   00E7'CF       80    0F68         ADDB2   DOTTEST,RF_FAB+FAB$B_FNS ; Keep a running count of spec length
63 00EF'CF   00E7'CF       28    0F6F         MOVC3   DOTTEST,DOTTEST+8,(R3)   ; Concatenate the file type
             1657'CF       9B    0F77         MOVZBW  RF_FAB+FAB$B_FNS,-           ; Save the length...
             1717'CF             0F7B                 RF_FILESPEC_DESC             ; ...in case we need it for error msg
                                 0F7E
             00F6'CF       90    0F7E         MOVB    SYSTEST_DIR,-                ; Set up a default directory
             1658'CF             0F82                 RF_FAB+FAB$B_DNS
             00FE'CF       9E    0F85         MOVAB   SYSTEST_DIR+8,-              ; This allows change without...
             1653'CF             0F89                 RF_FAB+FAB$L_DNA             ; ...having to re-form the filespec
        1633'CF     01     D0    0F8C         MOVL    #1,RF_FAB+FAB$L_ALQ          ; Get a minimum allocation
             50     01     D0    0F91         MOVL    #1,R0                        ; Indicate that we have a candidate
             58     68     D0    0F94         MOVL    UIDUCB$A_FLINK(R8),R8        ; Point to the next UCB on controller
 05  0F97  1985          RSB
```

```
                                  0F98  200$:                                               ; See if we can create a file
 00FF 8F    00    00 8F    00  2C  0F98         MOVC5    #0,#0,#0,#NAM$C_MAXRSS,-   ; Ensure that the result of any...
                     181E'CF       0FA0                  RESULT_FILESPEC             ; ...previous $CREATE is gone
                                  0FA3         $CREATE  FAB = RF_FAB                ; Make a file (we hope)
                       32 50   E8  0FAE         BLBS     R0,210$                     ; BR if we succeeded
      162B'CF     00000000'8F  D1  0FB1         CMPL     #RMS$_DNF,RF_FAB+FAB$L_STS ; Did we get directory not found?
                          36   12  0FBA         BNEQ     220$                        ; BR if not - we have no hopes
                     0107'CF   90  0FBC         MOVB     SYS0_SYSTEST_DIR,-         ; We did. Try for rooted directory...
                     1658'CF       0FC0                  RF_FAB+FAB$B_DNS
                     010F'CF   9E  0FC3         MOVAB    SYS0_SYSTEST_DIR+8,-       ; ...
                     1653'CF       0FC7                  RF_FAB+FAB$L_DNA
 00FF 8F    00    00 8F    00  2C  0FCA         MOVC5    #0,#0,#0,#NAM$C_MAXRSS,-   ; Ensure that the result of the...
                     181E'CF       0FD2                  RESULT_FILESPEC             ; ...previous $CREATE is gone
                                  0FD5         $CREATE  FAB = RF_FAB                ; Try again for the file
                       0F 50   E9  0FE0         BLBC     R0,220$                     ; Finish up with message if error
                                  0FE3  210$:
                                  0FE3         $CONNECT RAB = RF_RAB,-              ; Attach a RAB to our FAB
                                  0FE3                   ERR = RMS_ERROR
                                  0FF2  220$:
                          01   BB  0FF2         PUSHR    #^M<R0>                     ; Save RMS status
              51     0B60'CF   DE  0FF4         MOVAL    DEBUG_FILE_MSG,R1          ; Assume we created the file
                       05 50   E8  0FF9         BLBS     R0,230$                     ; BR if that was the case
              51     0B7D'CF   DE  0FFC         MOVAL    DEBUG_NOFILE_MSG,R1        ; Get a different message if not
                                  1001  230$:
              52     1717'CF   DE  1001         MOVAL    RF_FILESPEC_DESC,R2
                                  1006         $FAO_S   CTRSTR = (R1),-             ; Form a debugging message
                                  1006                  OUTLEN = DEBUG_PTR,-
                                  1006                  OUTBUF = DEBUG_FAO_BUF,-
                                  1006                  P1     = R2,-
                                  1006                  P2     = R0
                        0B8B   30  101B         BSBW     GIVE_DEBUG_MSG
                          01   BA  101E         POPR     #^M<R0>                     ; Restore RMS status
                               05  1020         RSB                                  ; Exit with the last RMS status in R0
```

```
                                   1021  300$:                                                ; Write and read a block of the file
      5A 8F    00 8F    00    2C   1021          MOVC5    #0,#0,#PATTERN_1,-                  ; Write some garbage...
       0CC4'CF    010D  8F         1027                   #TEXTB_SIZE,BUFFER
                                   102D          $PUT     RAB = RF_RAB,-                      ; ...to the file...
                                   102D                   ERR = RMS_ERROR
                  5F    50    E9   103C          BLBC     R0,320$
                                   103F          $REWIND  RAB = RF_RAB,-                      ; ...and see if...
                                   103F                   ERR = RMS_ERROR
                  4D    50    E9   104E          BLBC     R0,320$
                                   1051          $GET     RAB = RF_RAB,-                      ; ...we can reread it...
                                   1051                   ERR = RMS_ERROR
                  3B    50    E9   1060          BLBC     R0,320$
      5A 8F    00 8F    00    2D   1063          CMPC5    #0,#0,#PATTERN_1,-                  ; ...correctly
       0CC4'CF    010D  8F         1069                   #TEXTB_SIZE,BUFFER
                        2A    13   106F          BEQL     310$                                ; BR to clean exit
                  7E    63    9A   1071          MOVZBL   (R3),-(SP)                          ; Save the bad data...
            0000005A    8F    DD   1074          PUSHL    #PATTERN_1                          ; ...the good data...
  7E     0000010D 8F    52    C3   107A          SUBL3    R2,#TEXTB_SIZE,-(SP)                ; ...the offset of the bad data...
               1717'CF        DF   1082          PUSHAL   RF_FILESPEC_DESC                    ; ...the device...
            000F0004    8F    DD   1086          PUSHL    #^XF0004                            ; ...
            00748018    8F    DD   108C          PUSHL    #UETP$_DATADEVERR                   ; ...and the error code...
          1DAD'CF       06    FB   1092          CALLS    #6,ERROR_SIGNAL                     ; ...so we can warn of the error
                        50    D4   1097          CLRL     R0                                  ; Indicate that we had an error
                        03    11   1099          BRB      320$
                                   109B  310$:
                  50    01    D0   109B          MOVL     #1,R0                               ; Indicate success
                                   109E  320$:
                              05   109E          RSB
```

```
                               109F  400$:                                            ; Choose a slave to share file access
                               109F                                                   ; R1 returns an index for chosen node
                               109F  ;
                               109F  ; Use the filespec as the input to a hashing function so we can pick a
                               109F  ; "random" slave node for shared access.
                               109F  ;
          53   1717'CF    3C   109F              MOVZWL   RF_FILESPEC_DESC,R3        ; We will...
          54   171F'CF    DE   10A4              MOVAL    RF_FILESPEC,R4             ; ...
                               10A9  ;           CLRL     R1                         ; ...use a "random" seed...
                               10A9  410$:
             51    84     80   10A9              ADDB2    (R4)+,R1                   ; ...to sum the filespec chars
                  FA 53   F5   10AC              SOBGTR   R3,410$                    ; (Note that R3=0 when we fall thru)
                               10AF  ;           CLRL     R3                         ; Start counting assigned channels
                               10AF  420$:
              00AA'CF43   B5   10AF              TSTW     NODE_CHANS[R3]             ; Is this the first unassigned channel?
                     08   13   10B4              BEQL     430$                       ; We've finished counting, if so
 F1 53      000000FF 8F   F3   10B6              AOBLEQ   #MAX_NODES,R3,420$         ; Keep counting up to end of list
                               10BE  430$:
                     53   D5   10BE              TSTL     R3                         ; Have we any assigned channel?
                     20   13   10C0              BEQL     460$                       ; BR if not - no slave to share access
                     52   D4   10C2              CLRL     R2                         ; Set up for EDIV dividend operand
    51     51     51     53   7B   10C4              EDIV     R3,R1,R1,R1                ; Normalize "random" channel
                  54     51   D0   10C9              MOVL     R1,R4                      ; Prevent endless loop searching
                               10CC  440$:
       52    02AA'CF41    7E   10CC              MOVAQ    NODE_NAMES[R1],R2
                     01   E1   10D2              BBC      #CLIG_V_DEADNODE,-         ; BR if the slave is OK...
               2B 02 A2        10D4                       2(R2),470$                 ; ...to check shared access
             02 51   53   F2   10D7              AOBLSS   R3,R1,450$                 ; It's not, point to next possible slave
                     51   D4   10DB              CLRL     R1                         ; Wrap around if we're beyond valid ones
                               10DD  450$:
             54      51   D1   10DD              CMPL     R1,R4                      ; Have we an endless loop?
                     EA   12   10E0              BNEQ     440$                       ; BR if not - do further checks
                               10E2  460$:
       51    1717'CF      DE   10E2              MOVAL    RF_FILESPEC_DESC,R1        ; We're out of possible slaves...
                               10E7              $FAO_S   CTRSTR = DEBUG_NOSHARE_MSG,-
                               10E7                       OUTLEN = DEBUG_PTR,-
                               10E7                       OUTBUF = DEBUG_FAO_BUF,-
                               10E7                       P1     = R1
                   0AAA   30   10FC              BSBW     GIVE_DEBUG_MSG             ; ...let user know if debugging...
                     50   D4   10FF              CLRL     R0                         ; ...and indicate that we've failed
                          05   1101              RSB
                               1102  470$:
             50      01   D0   1102              MOVL     #1,R0                      ; Indicate that we have a candidate
                               1105                                                  ; R1 has the index of the slave
                          05   1105              RSB
```

```
                                  1106  500$:                                               ; Have a slave share access to a file
                           07C0   1106           .WORD    ^M<R6,R7,R8,R9,R10>              ; R2 through R5 may be trashed
                                  1108
             51    04 AC     D0   1108           MOVL     04(AP),R1                        ; Recall index for node to share access
       57    00AA'CF41       3E   110C           MOVAW    NODE_CHANS[R1],R7                ; Point to our DECnet channel
       58    02AA'CF41       7E   1112           MOVAQ    NODE_NAMES[R1],R8                ; Point to our node name
          59   0DE7'CF       DE   1118           MOVAL    ACCESS_MSG,R9                    ; Set up convenience registers...
          5A   0DEF'CF       DE   111D           MOVAL    CONTINUE_MSG,R10                 ; ...
    0AA2'CF    02 A9   69    28   1122           MOVC3    (R9),2(R9),MESSAGE_BUFFER        ; Set up message type
       50    010D 8F   69    A3   1129           SUBW3    (R9),#TEXTB_SIZE,R0              ; Figure space available for message
          51   1676'CF       9B   112F           MOVZBW   RF_NAM+NAM$B_RSL,R1              ; Figure length of filespec
                                  1134  ;        CMPW     R0,R1                            ; Have we enough room?
                                  1134  ;        BLSS                                      ; Should never be problem, by definition
          1677'DF    51      2C   1134           MOVC5    R1,@RF_NAM+NAM$L_RSA,-           ; Pass the filespec as our message
          63    50   00           1139                    #0,R0,(R3)
               7E    67      3C   113C           MOVZWL   (R7),-(SP)                       ; Set up the channel...
                     58      DD   113F           PUSHL    R8                               ; ...the node name...
                     59      DD   1141           PUSHL    R9                               ; ...and our message name
          1922'CF    03      FB   1143           CALLS    #3,MASTER_WRITE                  ; Tell this node to access our file
                                  1148           BLBCW    R0,550$                          ; Skip the rest if this node died
               7E    67      3C   114E           MOVZWL   (R7),-(SP)                       ; Set up the channel...
                     58      DD   1151           PUSHL    R8                               ; ...the node name...
                     59      DD   1153           PUSHL    R9                               ; ...and our message name
          19B0'CF    03      FB   1155           CALLS    #3,MASTER_READ                   ; See if the node got to our file
                                  115A           BLBCW    R0,550$                          ; Some error, skip the rest
   0CC4'CF     02 A9   69    29   1160           CMPC3    (R9),2(R9),BUFFER                ; Did we get the reply we expected?
                     16      13   1167           BEQL     510$                             ; BR if we did
             0999'CF         DF   1169           PUSHAL   EXCLUDE_MSG                      ; Complain if we did not
                     58      DD   116D           PUSHL    R8
                     59      DD   116F           PUSHL    R9
          1B47'CF    03      FB   1171           CALLS    #3,GARBLED_TRANS
             02 A8   02      A8   1176           BISW2    #CLIG_M_DEADNODE,2(R8)           ; Mark the node as unuseable
                     50      D4   117A           CLRL     R0                               ; Indicate that we failed
                   0107      31   117C           BRW      550$                             ; Skip the rest - node is incoherent
                                  117F  510$:
                  49 63      E8   117F           BLBS     (R3),520$                        ; BR if node could access the file
                     63      DD   1182           PUSHL    (R3)                             ; Otherwise get the error status
          1BC3'CF    01      FB   1184           CALLS    #1,STATUS_TO_TEXT                ; Convert it to something we can type
          54   1717'CF       7E   1189           MOVAQ    RF_FILESPEC_DESC,R4
                                  118E           $FAO_S   CTRSTR = SLAVE_NO_ACCESS,- ; Tell the user what happened
                                  118E                    OUTLEN = BUFFER_PTR,-
                                  118E                    OUTBUF = FAO_BUF,-
                                  118E                    P1     = R8,-
                                  118E                    P2     = R4
             0EDE'CF         DF   11A5           PUSHAL   STATUS_PTR
                     01      DD   11A9           PUSHL    #1
          00741132 8F        DD   11AB           PUSHL    #UETP$_TEXT!STS$K_ERROR
             0CBC'CF         DF   11B1           PUSHAL   BUFFER_PTR
          000F0001 8F        DD   11B5           PUSHL    #^XF0001
          00741132 8F        DD   11BB           PUSHL    #UETP$_TEXT!STS$K_ERROR
          1DAD'CF    06      FB   11C1           CALLS    #6,ERROR_SIGNAL
                     50      D4   11C6           CLRL     R0                               ; Indicate a failure
                   00BB      31   11C8           BRW      550$                             ; Skip the rest for this file
                                  11CB  520$:
     F0 8F     00 8F   00    2C   11CB           MOVC5    #0,#0,#PATTERN_2,-               ; Set up a second record for the file
     0CC4'CF    010D 8F           11D1                    #TEXTB_SIZE,BUFFER
                                  11D7           $PUT     RAB = RF_RAB,-                   ; Write that garbage, too
```

```
                               11D7                       ERR = RMS_ERROR
                               11E6       ;      BLBC     R0,550$                      ; No point in checking errors - ...
                               11E6                                                    ; ...the slave must try to read
                               11E6              $FLUSH   RAB = RF_RAB,-               ; Ensure that it gets out to our file
                               11E6                       ERR = RMS_ERROR
                               11F5       ;      BLBC     R0,550$                      ; No point in checking errors - ...
                               11F5                                                    ; ...the slave must try to read
 0AA2'CF    02 AA    6A    28  11F5              MOVC3    (R10),2(R10),MESSAGE_BUFFER ; Tell slave to read the next block
               7E    67    3C  11FC              MOVZWL   (R7),-(SP)                   ; Set up the channel...
                     58    DD  11FF              PUSHL    R8                           ; ...the node name...
                     5A    DD  1201              PUSHL    R10                          ; ...and our message name
         1922'CF     03    FB  1203              CALLS    #3,MASTER_WRITE              ; Tell the slave to read second block
                 7B  50    E9  1208              BLBC     R0,550$                      ; Skip the rest if there's an error
               7E    67    3C  120B              MOVZWL   (R7),-(SP)                   ; Set up the channel...
                     58    DD  120E              PUSHL    R8                           ; ...the node name...
                     5A    DD  1210              PUSHL    R10                          ; ...and our message name
         19B0'CF     03    FB  1212              CALLS    #3,MASTER_READ               ; See if slave read second block
                 6C  50    E9  1217              BLBC     R0,550$                      ; BR if slave had trouble
 0CC4'CF    02 AA    6A    29  121A              CMPC3    (R10),2(R10),BUFFER          ; Did we get the reply we expected?
                     15    13  1221              BEQL     530$                         ; BR if we did
            0999'CF        DF  1223              PUSHAL   EXCLUDE_MSG                  ; Complain if we did not
                     58    DD  1227              PUSHL    R8
                     5A    DD  1229              PUSHL    R10
         1B47'CF     03    FB  122B              CALLS    #3,GARBLED_TRANS
           02 A8     02    A8  1230              BISW2    #CLIG_M_DEADNODE,2(R8)       ; Mark the node as unuseable
                     50    D4  1234              CLRL     R0                           ; Indicate that we failed
                     4E    11  1236              BRB      550$                         ; Skip the rest - node is incoherent
                               1238       530$:
               48    63    E8  1238              BLBS     (R3),540$                    ; BR if node could read extended file
                     63    DD  123B              PUSHL    (R3)                         ; Otherwise get the error status
         1BC3'CF     01    FB  123D              CALLS    #1,STATUS_TO_TEXT            ; Convert it to something we can type
         54   1717'CF      7E  1242              MOVAQ    RF_FILESPEC_DESC,R4
                               1247              $FAO_S   CTRSTR = SLAVE_EXT_FAIL,-    ; Tell the user what happened
                               1247                       OUTLEN = BUFFER_PTR,-
                               1247                       OUTBUF = FAO_BUF,-
                               1247                       P1     = R8,-
                               1247                       P2     = R4
            0EDE'CF        DF  125E              PUSHAL   STATUS_PTR
                     01    DD  1262              PUSHL    #1
        00741132     8F    DD  1264              PUSHL    #UETP$_TEXT!STS$K_ERROR
            0CBC'CF        DF  126A              PUSHAL   BUFFER_PTR
        000F0001     8F    DD  126E              PUSHL    #^XF0001
        00741132     8F    DD  1274              PUSHL    #UETP$_TEXT!STS$K_ERROR
         1DAD'CF     06    FB  127A              CALLS    #6,ERROR_SIGNAL
                     50    D4  127F              CLRL     R0                           ; Indicate a failure
                     03    11  1281              BRB      550$                         ; Skip the rest for this file
                               1283       540$:
              50     01    D0  1283              MOVL     #1,R0                        ; Indicate success
                               1286       550$:
                           04  1286              RET                                   ; That's it for shared access
```

```
                                 1287  600$:
              57   00AA'CF   3E  1287          MOVAW   NODE_CHANS,R7              ; Tell all slaves to end file access
              58   02AA'CF   7E  128C          MOVAQ   NODE_NAMES,R8              ; Used to loop through DECnet channels
              59   0DF9'CF   DE  1291          MOVAL   MOVE_ON_MSG,R9             ; Used to loop through node name descs
   0AA2'CF    02 A9     69   28  1296          MOVC3   (R9),2(R9),MESSAGE_BUFFER  ; Set up convenience register
                                 129D  610$:                                      ; Set up message
                        67   B5  129D          TSTW    (R7)                       ; Have we another channel?
                        01   12  129F          BNEQ    620$                       ; BR if so - tell node to move on
                             05  12A1          RSB
                                 12A2  620$:
                   7E   87   3C  12A2          MOVZWL  (R7)+,-(SP)                ; Set up channel (and point to next)...
                        58   DD  12A5          PUSHL   R8                         ; ...the node name...
                        59   DD  12A7          PUSHL   R9                         ; ...and our message
              1922'CF   03   FB  12A9          CALLS   #3,MASTER_WRITE            ; Tell node to move on after file access
                        88   73  12AE          TSTD    (R8)+                      ; Point to the next possible name desc
                        EB   11  12B0          BRB     610$                       ; Loop for the next node
```

```
                              12B2          .SBTTL  SHARE_ACCESS - See If We can Share File Access
                              12B2     ;++
                              12B2     ; FUNCTIONAL DESCRIPTION:
                              12B2     ;        See if a slave can read a file or files that is being written by the
                              12B2     ;        master process.
                              12B2     ;
                              12B2     ; IMPLICIT INPUTS:
                              12B2     ;        Name of a file, by way of a message from the master process.
                              12B2     ;
                              12B2     ; IMPLICIT OUTPUTS:
                              12B2     ;        NONE
                              12B2     ;
                              12B2     ; SIDE EFFECTS:
                              12B2     ;        File is read and deaccessed.
                              12B2     ;
                              12B2     ;--
                              12B2
                              12B2    SHARE_ACCESS:
            59  0DE7'CF    DE 12B2            MOVAL   ACCESS_MSG,R9              ; Set up convenience registers...
            5A  0DEF'CF    DE 12B7            MOVAL   CONTINUE_MSG,R10           ; ...
            5B  0DF9'CF    DE 12BC            MOVAL   MOVE_ON_MSG,R11            ; ...
                              12C1    10$:
                       59  DD 12C1            PUSHL   R9                         ; Define the type of message we expect
            16D0'CF    01  FB 12C3            CALLS   #1,SLAVE_READ              ; Get the master node's message
    0AA2'CF    02 A9   69  29 12C8            CMPC3   (R9),2(R9),MESSAGE_BUFFER ; What does the message say?
                       31  13 12CF            BEQL    30$                        ; BR if we're to access a file
    0AA2'CF    02 AB   6B  29 12D1            CMPC3   (R11),2(R11),MESSAGE_BUFFER ; Are we done with this section?
                       1C  13 12D8            BEQL    20$                        ; BR if so
                  00BB'CF  DF 12DA            PUSHAL  NULL                       ; Otherwise...
                  0094'CF  DF 12DE            PUSHAL  MASTER_NODE_DESC
                       59  DD 12E2            PUSHL   R9                         ; ...we're confused...
            1B47'CF    03  FB 12E4            CALLS   #3,GARBLED_TRANS           ; ...and can't do anything about it
                              12E9            $EXIT_S CODE = #UETP$_ABENDD!STS$K_ERROR!STS$M_INHIB_MSG
                              12F6    20$:
                              12F6            $CLOSE  FAB = RF_FAB               ; Blindly deaccess any possible file
                           05 1301            RSB
                              1302    30$:
            63  00FF 8F    28 1302            MOVC3   #NAM$C_MAXRSS,(R3),-       ; Set up the filespec - name...
                  171F'CF     1307                    RF_FILESPEC
            00FF 8F    00  3A 130A            LOCC    #0,#NAM$C_MAXRSS,-         ; ...
                  171F'CF     130F                    RF_FILESPEC
            00FF 8F    50  A3 1312            SUBW3   R0,#NAM$C_MAXRSS,-         ; ...and length
                  1717'CF     1317                    RF_FILESPEC_DESC
                  1717'CF  90 131A            MOVB    RF_FILESPEC_DESC,-         ; Set the length...
                  1657'CF     131E                    RF_FAB+FAB$B_FNS           ; ...where RMS expects it
00FF 8F    00    00 8F    00  2C 1321            MOVC5   #0,#0,#0,#NAM$C_MAXRSS,- ; Clear out remnants...
                  181E'CF     1329                    RESULT_FILESPEC            ; ...of any previous $OPEN...
                       01  8A 132C            BICB    #FAB$M_PUT,-               ; ...and be honest about our access
                  1639'CF     132E                    RF_FAB+FAB$B_FAC
                              1331            $OPEN   FAB = RF_FAB,-             ; See if we can get to the file
                              1331                    ERR = RMS_ERROR
                              1340            BLBCW   R0,40$                     ; Skip the rest if we get an error
            50  0042'CF    DE 1346            MOVAL   SCSNODE,R0
            51  1717'CF    DE 134B            MOVAL   RF_FILESPEC_DESC,R1
                              1350            $FAO_S  CTRSTR = DEBUG_SHARE_MSG,- ; If we're tracing, say...
                              1350                    OUTLEN = DEBUG_PTR,-
                              1350                    OUTBUF = DEBUG_FAO_BUF,-
                              1350                P1      = #NODE_LENGTH,-
                              1350                P2      = R0,-
                              1350                P3      = R1
               083D    30     1369         BSBW    GIVE_DEBUG_MSG          ; ...that we've gotten to the file
                              136C         $CONNECT RAB = RF_RAB,-
                              136C                 ERR = RMS_ERROR
              4B 50    E9     137B         BLBC    R0,40$                  ; Skip the rest if we get an error
                              137E         $GET    RAB = RF_RAB,-          ; Try to read the file
                              137E                 ERR = RMS_ERROR
              39 50    E9     138D         BLBC    R0,40$                  ; Skip the rest if we get an error
     5A 8F    00 8F    00 2D  1390         CMPC5   #0,#0,#PATTERN_1,-      ; Did we read the correct data?
      0CC4'CF   010D 8F       1396                 #TEXTB_SIZE,BUFFER
                 45    13     139C         BEQL    50$                     ; BR if we did
              7E 63    9A     139E         MOVZBL  (R3),-(SP)              ; Save the bad data...
          7E   5A 8F   9A     13A1         MOVZBL  #PATTERN_1,-(SP)        ; ...the good data...
   7E   0000010D 8F    52 C3  13A5         SUBL3   R2,#TEXTB_SIZE,-(SP)    ; ...the offset of the bad data...
               1717'CF DF     13AD         PUSHAL  RF_FILESPEC_DESC        ; ...the device...
           000F0004 8F DD     13B1         PUSHL   #^XF0004                ; ...
           00748018 8F DD     13B7         PUSHL   #UETP$_DATADEVERR       ; ...and the error code...
          1DAD'CF  06  FB     13BD         CALLS   #6,ERROR_SIGNAL         ; ...so we can indicate the problem...
     50   00748018 8F  D0     13C2         MOVL    #UETP$_DATADEVERR,R0    ; ...and warn of the error
                              13C9  40$:
        0AA8'CF    50  D0     13C9         MOVL    R0,MESSAGE_BUFFER+-     ; Use our error code as a message
                              13CE                 ACCESS_LENGTH
                              13CE         $CLOSE  FAB = RF_FAB            ; Deaccess this file
                   59  DD     13D9         PUSHL   R9                      ; Save the type of message...
        1769'CF    01  FB     13DB         CALLS   #1,SLAVE_WRITE          ; ...and tell master we had problems
                 FEDE  31     13E0         BRW     10$
                              13E3  50$:
        0AA8'CF    01  D0     13E3         MOVL    #1,MESSAGE_BUFFER+-     ; Reply to master - MESSAGE_BUFFER...
                              13E8                 ACCESS_LENGTH
                   59  DD     13E8         PUSHL   R9                      ; ...still has correct message type...
        1769'CF    01  FB     13EA         CALLS   #1,SLAVE_WRITE          ; ...to which we append success
                   5A  DD     13EF         PUSHL   R10                     ; Define the type of message we want
        16D0'CF    01  FB     13F1         CALLS   #1,SLAVE_READ           ; Let master tell us to read next block
  0AA2'CF   02 AA  6A  29     13F6         CMPC3   (R10),2(R10),MESSAGE_BUFFER ; What does the message say?
                   31  13     13FD         BEQL    70$                     ; BR if we're to continue access
  0AA2'CF   02 AB  6B  29     13FF         CMPC3   (R11),2(R11),MESSAGE_BUFFER ; Did master tell us to move on?
                   1C  13     1406         BEQL    60$                     ; BR if so - clean up
            00BB'CF    DF     1408         PUSHAL  NULL                    ; Otherwise...
            0094'CF    DF     140C         PUSHAL  MASTER_NODE_DESC
                   5A  DD     1410         PUSHL   R10                     ; ...we're confused...
        1B47'CF    03  FB     1412         CALLS   #3,GARBLED_TRANS        ; ...and can't do anything about it
                              1417         $EXIT_S CODE = #UETP$_ABENDD!STS$K_ERROR!STS$M_INHIB_MSG
                              1424  60$:
                              1424         $CLOSE  FAB = RF_FAB            ; Get out as easily as possible
                   05         142F         RSB
                              1430  70$:
                              1430         $CLOSE  FAB = RF_FAB,-
                              1430                 ERR = RMS_ERROR
                              143F         BLBCW   R0,80$                  ; Skip the rest if we get an error
                              1445         $OPEN   FAB = RF_FAB,-          ; Update our knowledge of the file
                              1445                 ERR = RMS_ERROR
              6F 50    E9     1454         BLBC    R0,80$                  ; Skip the rest if we get an error
                              1457         $CONNECT RAB = RF_RAB,-
                              1457                 ERR = RMS_ERROR
              5D 50    E9     1466         BLBC    R0,80$                  ; Skip the rest if we get an error
```

```
                               1469        $GET    RAB = RF_RAB,-              ; Reread the first record
                               1469                ERR = RMS_ERROR
                   4B 50   E9  1478        BLBC    R0,80$                      ; Skip the rest if we get an error
                               147B        $GET    RAB = RF_RAB,-              ; Try to read a second record
                               147B                ERR = RMS_ERROR
                   39 50   E9  148A        BLBC    R0,80$                      ; Skip the rest if we get an error
        F0 8F    00 8F 00  2D  148D        CMPC5   #0,#0,#PATTERN_2,-          ; Did we read the correct data?
       0CC4'CF   010D 8F       1493                #TEXTB_SIZE,BUFFER
                      2B   13  1499        BEQL    80$                         ; BR if we did - note that R0 = 0
                   7E 63   9A  149B        MOVZBL  (R3),-(SP)                  ; Save the bad data...
                 7E  F0 8F 9A  149E        MOVZBL  #PATTERN_2,-(SP)            ; ...the good data...
    7E   0000010D 8F  52   C3  14A2        SUBL3   R2,#TEXTB_SIZE,-(SP)        ; ...the offset of the bad data...
                 1717'CF   DF  14AA        PUSHAL  RF_FILESPEC_DESC            ; ...the "device"...
              000F0004 8F  DD  14AE        PUSHL   #^XF0004                    ; ...
              00748018 8F  DD  14B4        PUSHL   #UETP$_DATADEVERR           ; ...and the error code...
              1DAD'CF  06  FB  14BA        CALLS   #6,ERROR_SIGNAL             ; ...so we can indicate the problem...
          50  00748018 8F  D0  14BF        MOVL    #UETP$_DATADEVERR,R0        ; ...and warn of the error
                               14C6  80$:
                      50   D5  14C6        TSTL    R0                          ; R0 = 0 if all OK, else error code
                      29   12  14C8        BNEQ    90$                         ; BR if we had a problem
             50   0042'CF  DE  14CA        MOVAL   SCSNODE,R0
             51   1717'CF  DE  14CF        MOVAL   RF_FILESPEC_DESC,R1
                               14D4        $FAO_S  CTRSTR = DEBUG_EXTEND_MSG,-
                               14D4                OUTLEN = DEBUG_PTR,-
                               14D4                OUTBUF = DEBUG_FAO_BUF,-
                               14D4                P1     = #NODE_LENGTH,-
                               14D4                P2     = R0,-
                               14D4                P3     = R1
                    06B9   30  14ED        BSBW    GIVE_DEBUG_MSG              ; Let debugging user know...
                 50   01   D0  14F0        MOVL    #1,R0                       ; ...that we read the extended file
                               14F3  90$:
             0AAA'CF  50   D0  14F3        MOVL    R0,MESSAGE_BUFFER+-         ; Use status code as our message
                               14F8                CONTINUE_LENGTH
                               14F8        $CLOSE  FAB = RF_FAB                ; We've accessed the file
                               1503  ;             ERR = RMS_ERROR             ; Get here on error as well as success
                      5A   DD  1503        PUSHL   R10                         ; Message says we're finished with file
             1769'CF  01   FB  1505        CALLS   #1,SLAVE_WRITE              ; Return result of sharing access
                    FDB4   31  150A        BRW     10$                         ; Loop in case we have to do another
```

```
                                  150D          .SBTTL  WIND_DOWN - Terminate Slaves and Clean Up
                                  150D   ;++
                                  150D   ; FUNCTIONAL DESCRIPTION:
                                  150D   ;       Allow the slave processes to exit.  Each of the slave processes will
                                  150D   ;       relay its copy of SYS$ERROR.LOG back to us; we will copy the relevant
                                  150D   ;       parts of it to our own SYS$OUTPUT.  Announce the end of testing to
                                  150D   ;       the operators' consoles in the cluster.
                                  150D   ;
                                  150D   ; IMPLICIT INPUTS:
                                  150D   ;       NODE_CHAN list of channels on which we have DECnet links
                                  150D   ;
                                  150D   ; IMPLICIT OUTPUTS:
                                  150D   ;       NONE
                                  150D   ;
                                  150D   ; SIDE EFFECTS:
                                  150D   ;       DECnet tasks are terminated.
                                  150D   ;       Slave SYS$ERROR files copied to our SYS$OUTPUT.
                                  150D   ;       Message to various operator consoles.
                                  150D   ;
                                  150D   ;--
                                  150D
                                  150D   WIND_DOWN:
             57   00AA'CF    3E   150D          MOVAW   NODE_CHANS,R7             ; Used to loop through DECnet channels
             58   02AA'CF    7E   1512          MOVAQ   NODE_NAMES,R8             ; Used to loop through node name descs
             5A   0E02'CF    DE   1517          MOVAL   ERRORLOG_MSG,R10          ; Set up convenience registers...
             59   0E0C'CF    DE   151C          MOVAL   ERRORLOG_ENDED_MSG,R9     ; ...
                                  1521   10$:
                          67 B5   1521          TSTW    (R7)                      ; Have we another channel?
                                  1523          BEQLW   40$                       ; BR if not - all SYS$ERROR.LOGs copied
                                  1528
                                  1528          $PUTMSG_S MSGVEC = BLANK_LINE_PTR ; Set off logs with a blank line
                          58 DD   1539          PUSHL   R8                        ; Set up a message...
                          01 DD   153B          PUSHL   #1                        ; ...
             007480B1 8F     DD   153D          PUSHL   #UETP$_COPY_LOG
             000F0003 8F     DD   1543          PUSHL   #^XF0003
                   50     5E DO   1549          MOVL    SP,R0
                                  154C          $PUTMSG_S MSGVEC = (R0)   ; ...which log we're copying
                          0F BA   155B          POPR    #^M<R0,R1,R2,R3>          ; Clean MSGVEC from the stack
                                  155D   20$:
                   7E     67 3C   155D          MOVZWL  (R7),-(SP)                ; Set up the channel...
                          58 DD   1560          PUSHL   R8                        ; ...the node name...
                          5A DD   1562          PUSHL   R10                       ; ...and our message name
             1A3E'CF      03 FB   1564          CALLS   #3,MASTER_ERRORLOG_READ   ; Get a slave's non-success message
                   4A     50 E9   1569          BLBC    R0,30$                    ; Give up if an error
   0CC4'CF     02 A9      69 29   156C          CMPC3   (R9),2(R9),BUFFER         ; Is it an ERRORLOG_ENDED message?
                          41 13   1573          BEQL    30$                       ; BR if so - we've finished this slave
   0CC4'CF     02 AA      6A 29   1575          CMPC3   (R10),2(R10),BUFFER       ; Is it an ERRORLOG message?
                          DF 12   157C          BNEQ    20$                       ; BR if not - we're out of synch
          021A 8F      00    3A   157E          LOCC    #0,#2*TEXTB_SIZE,-        ; Find the end of the message
                 0CCC'CF          1583                  BUFFER+ERRORLOG_LENGTH
      0000021A 8F      50    C3   1586          SUBL3   R0,#2*TEXTB_SIZE,-        ; Use it to compute the message length
                 0CBC'CF          158D                  BUFFER_PTR
                          CB 13   1590          BEQL    20$                       ; Don't print slave's empty message
                 0CCC'CF     DE   1592          MOVAL   BUFFER+ERRORLOG_LENGTH,-  ; Point past the message type...
                 0CC0'CF          1596                  BUFFER_PTR+4              ; ...so that the message is clear
                    00E4     30   1599          BSBW    100$                      ; Indent the line(s) of the message
                                  159C          $PUTMSG_S MSGVEC = ERRORLOG_PTR   ; Copy slave SYS$ERROR to our SYS$OUTPUT
```

```
 0CC0'CF    0CC4'CF   DE  15AD              MOVAL    BUFFER,BUFFER_PTR+4        ; Reset buffer pointer to buffer's start
                 A7   11  15B4              BRB      20$                        ; Loop for the next message
                          15B6        30$:
                 58   DD  15B6              PUSHL    R8                         ; Set up a message...
                 01   DD  15B8              PUSHL    #1                         ; ...
        007480C1 8F   DD  15BA              PUSHL    #UETP$_COPY_LOG_ENDED
        000F0003 8F   DD  15C0              PUSHL    #^XF0003                   ; ...to say...
           50    5E   D0  15C6              MOVL     SP,R0
                          15C9              $PUTMSG_S MSGVEC = (R0)             ; ...which log we've copied
                 0F   BA  15D8              POPR     #^M<R0,R1,R2,R3>           ; Clean MSGVEC from the stack
                 87   B5  15DA              TSTW     (R7)+                      ; Point to the next possible channel
                 88   73  15DC              TSTD     (R8)+                      ; Point to the next possible name desc
               FF40   31  15DE              BRW      10$                        ; Loop for the next slave's SYS$ERROR
                          15E1        40$:
      50    0042'CF   DE  15E1              MOVAL    SCSNODE,R0
                          15E6              $FAO_S   CTRSTR = END_OF_TESTING,-
                          15E6                       OUTLEN = BUFFER_PTR,-
                          15E6                       OUTBUF = FAO_BUF,-
                          15E6                       P1     = #NODE_LENGTH,-
                          15E6                       P2     = R0,-
                          15E6                       P3     = #0
                          15FF              $BRKTHRUW_S -                       ; Warn other nodes by a console message
                          15FF                       MSGBUF = BUFFER_PTR,-
                          15FF                       EFN    = #SS_SYNCH_EFN,-
                          15FF                       SENDTO = OPA0,-
                          15FF                       SNDTYP = #BRK$C_DEVICE,-
                          15FF                       FLAGS  = #BRK$M_CLUSTER,-
                          15FF                       TIMOUT = #BRKTHRU_TIMOUT,-
                          15FF                       IOSB   = QUAD_STATUS
        0A  002C'CF   E9  1624              BLBC     QUAD_STATUS,50$            ; BR if there was any error in sending
            0030'CF   A1  1629              ADDW3    QUAD_STATUS+4,-            ; Did all nodes see the warning?
      51    0032'CF       162D                       QUAD_STATUS+6,R1
                 4C   13  1631              BEQL     60$                        ; Skip the message if so
                          1633        50$:
      7E    002C'CF   3C  1633              MOVZWL   QUAD_STATUS,-(SP)          ; Get the text...
      1BC3'CF    01   FB  1638              CALLS    #1,STATUS_TO_TEXT          ; ...associated with any error
      51    0030'CF   3C  163D              MOVZWL   QUAD_STATUS+4,R1
      52    0032'CF   3C  1642              MOVZWL   QUAD_STATUS+6,R2
                          1647              $FAO_S   CTRSTR = BRKTHRU_ERRORS,- ; Form a message
                          1647                       OUTLEN = BUFFER_PTR,-
                          1647                       OUTBUF = FAO_BUF,-
                          1647                       P1     = R1,-
                          1647                       P2     = R2
            0EDE'CF   DF  165E              PUSHAL   STATUS_PTR
                 01   DD  1662              PUSHL    #1
        00741132 8F   DD  1664              PUSHL    #UETP$_TEXT!STS$K_ERROR
            0CBC'CF   DF  166A              PUSHAL   BUFFER_PTR
        000F0001 8F   DD  166E              PUSHL    #^XF0001
        00741132 8F   DD  1674              PUSHL    #UETP$_TEXT!STS$K_ERROR
      1DAD'CF    06   FB  167A              CALLS    #6,ERROR_SIGNAL
                          167F        60$:
                 05       167F              RSB
```

```
                                   1680  ;
                                   1680  ; Massage a record from the slave's SYS$ERROR file so that it is uniformly
                                   1680  ; indented from the left margin, even if the record contains embedded carriage
                                   1680  ; returns, line feeds and tabs.
                                   1680  ;
                                   1680  100$:
          51   0CC0'CF     D0      1680          MOVL    BUFFER_PTR+4,R1          ; R1 and R0 are a string desc...
          50   0CBC'CF     3C      1685          MOVZWL  BUFFER_PTR,R0            ; ...for the remainder of the record
                    7E  50 B0      168A          MOVW    R0,-(SP)                 ; Counts chars as indentation is done
                        1E 11      168D          BRB     130$                     ; BR inside loop - indent string's start
                                   168F  110$:
          61   50   0D     3A      168F          LOCC    #13,R0,(R1)              ; Is there a <RET> in rest of string?
                        35 13      1693          BEQL    140$                     ; Exit loop if not - no more indent
                        50 D7      1695          DECL    R0                       ; Found one. LOCC has us pointing at it
                        51 D6      1697          INCL    R1                       ; Point past the <RET>
               61   0A     91      1699          CMPB    #10,(R1)                 ; Is there a <LINEFEED>?
                        04 12      169C          BNEQ    120$                     ; BR if we need not skip <LINEFEED>
                        50 D7      169E          DECL    R0                       ; Must pass over <LF>...
                        51 D6      16A0          INCL    R1                       ; ...since they're new line to printers
                                   16A2  120$:
               61   09     91      16A2          CMPB    #9,(R1)                  ; Is there a tab at start of line?
                        06 12      16A5          BNEQ    130$                     ; BR if not - we can start indenting
                        50 D7      16A7          DECL    R0                       ; Must pass over the tab
                        51 D6      16A9          INCL    R1                       ; More of passing over the tab
                        F5 11      16AB          BRB     120$                     ; Inner loop to find multiple tabs
                                   16AD  130$:
                        50 D5      16AD          TSTL    R0                       ; If we're at the end of the string...
                        19 13      16AF          BEQL    140$                     ; ...we can exit the outer loop
                        03 BB      16B1          PUSHR   #^M<R0,R1>               ; Save desc to rest of string
          04 A1   61   50  28      16B3          MOVC3   R0,(R1),INDENT(R1)       ; Indent the rest of the string
04 BE   04   20   00 8F    00 2C   16B8          MOVC5   #0,#0,#^A/ /,#INDENT,@4(SP) ; Fill indented spaces with blanks
                        03 BA      16C0          POPR    #^M<R0,R1>               ; Restore desc to rest of string
               51       04 C0      16C2          ADDL2   #INDENT,R1               ; Point beyond the spaces just inserted
               6E       04 A0      16C5          ADDW2   #INDENT,(SP)             ; Count total length incl. indentation
                        C5 11      16C8          BRB     110$                     ; Loop to see if we need indent again
                                   16CA  140$:
          0CBC'CF       8E B0      16CA          MOVW    (SP)+,BUFFER_PTR         ; Set new record size
                           05      16CF          RSB                              ; Return with finished record
```

```
                  16D0        .SBTTL  Read and Write DECnet
                  16D0  ;++
                  16D0  ; FUNCTIONAL DESCRIPTION:
                  16D0  ;       A set of common routines to read from and write to DECnet.  They handle
                  16D0  ;       master and slave reading and writing as well as minimal error checking.
                  16D0  ;
                  16D0  ; CALLING SEQUENCE:
                  16D0  ;       CALLS   #3,MASTER_access
                  16D0  ;               - or -
                  16D0  ;       CALLS   #1,SLAVE_access
                  16D0  ;               and access is either READ or WRITE
                  16D0  ;
                  16D0  ; INPUT PARAMETERS:
                  16D0  ;       04(AP)  address of MESSAGE_NAMES message (count word followed by text)
                  16D0  ;       08(AP)  address of node name (master routines only)
                  16D0  ;       12(AP)  DECnet channel (master routines only)
                  16D0  ;
                  16D0  ; IMPLICIT INPUTS:
                  16D0  ;       NODE_CHANS has the DECnet channel (slave routines only)
                  16D0  ;       MESSAGE_BUFFER has the message to write (write routines only)
                  16D0  ;
                  16D0  ; OUTPUT PARAMETERS:
                  16D0  ;       NONE
                  16D0  ;
                  16D0  ; IMPLICIT OUTPUTS:
                  16D0  ;       QUAD_STATUS receives the status of the operation
                  16D0  ;       MESSAGE_BUFFER receives the message (slave read routine only)
                  16D0  ;       BUFFER receives the message (master read routine only)
                  16D0  ;
                  16D0  ; COMPLETION CODES:
                  16D0  ;       I/O status block status from $QIO
                  16D0  ;
                  16D0  ; SIDE EFFECTS:
                  16D0  ;       DECnet read or written
                  16D0  ;       Node no longer accessible (master routines only)
                  16D0  ;       Error message if there were problems
                  16D0  ;       Slave process may also exit if problems
                  16D0  ;
                  16D0  ;--
                  16D0
                  16D0  SLAVE_READ:
            0004  16D0          .WORD   ^M<R2>
                  16D2
                  16D2          $SETIMR_S DAYTIM = SLAVE_QIO_DELTA,- ; Prevent hangs waiting for DECnet
                  16D2                  ASTADR = TIME_OUT,-
                  16D2                  REQIDT = AP
                  16E5          $QIOW_S EFN    = #SS_SYNCH_EFN,- ; Get the master node's message
                  16E5                  CHAN   = NODE_CHANS,-
                  16E5                  FUNC   = #IO$_READVBLK,-
                  16E5                  IOSB   = QUAD_STATUS,-
                  16E5                  P1     = MESSAGE_BUFFER,-
                  16E5                  P2     = #TEXTB_SIZE
                  170A          $CANTIM_S REQIDT = AP           ; We returned from the DECnet QIO
1D 002C'CF    E8  1715          BLBS    QUAD_STATUS,10$         ; BR if message received correctly
   00B8'CF    DF  171A          PUSHAL  NULL                    ; Otherwise,...
   0094'CF    DF  171E          PUSHAL  MASTER_NODE_DESC
     04 AC    DD  1722          PUSHL   04(AP)
```

```
 1B29'CF    03   FB  1725        CALLS   #3,READ_FAILED              ; ...signal the error
                     172A        $EXIT_S CODE = #UETP$_ABENDD!STS$K_ERROR!STS$M_INHIB_MSG
                     1737   10$:
   50    04 AC   D0  1737        MOVL    04(AP),R0                   ; Point to the message
        51    60 3C  173B        MOVZWL  (R0),R1                     ; Get the message length
   50    02 A0   DE  173E        MOVAL   2(R0),R0                    ; Point to the message text
52    0094'CF    DE  1742        MOVAL   MASTER_NODE_DESC,R2
                     1747        $FAO_S  CTRSTR = DEBUG_READ_MSG,- ; Form debug message
                     1747                OUTLEN = DEBUG_PTR,-
                     1747                OUTBUF = DEBUG_FAO_BUF,-
                     1747                P1     = R1,-
                     1747                P2     = R0,-
                     1747                P3     = R2
          0446   30  1760        BSBW    GIVE_DEBUG_MSG              ; Let a debugging user see it
50    002C'CF    3C  1763        MOVZWL  QUAD_STATUS,R0              ; Return $QIO result
                 04  1768        RET
```

```
                          1769  ;+
                          1769  ;       One of the DECnet read/write routines.
                          1769  ;-
                          1769  SLAVE_WRITE:
                  0004    1769          .WORD   ^M<R2>
                          176B
                          176B          $SETIMR_S DAYTIM = SLAVE_QIO_DELTA,- ; Prevent hangs waiting for DECnet
                          176B                  ASTADR = TIME_OUT,-
                          176B                  REQIDT = AP
                          177E          $QIOW_S EFN     = #SS_SYNCH_EFN,- ; Answer the master node's message
                          177E                  CHAN    = NODE_CHANS,-
                          177E                  FUNC    = #IO$_WRITEVBLK,-
                          177E                  IOSB    = QUAD_STATUS,-
                          177E                  P1      = MESSAGE_BUFFER,-
                          177E                  P2      = #TEXTB_SIZE
                          17A3          $CANTIM_S REQIDT = AP           ; We returned from the DECnet QIO
     1D 002C'CF      E8   17AE          BLBS    QUAD_STATUS,10$         ; BR if message was sent correctly
        00F3'CF      DF   17B3          PUSHAL  NULL                    ; Otherwise...
        0094'CF      DF   17B7          PUSHAL  MASTER_NODE_DESC
          04 AC      DD   17BB          PUSHL   04(AP)
  1B38'CF    03      FB   17BE          CALLS   #3,WRITE_FAILED
                          17C3          $EXIT_S CODE = #UETP$_ABENDD!STS$K_ERROR!STS$M_INHIB_MSG
                          17D0  10$:
  50      04 AC      D0   17D0          MOVL    04(AP),R0               ; Point to the message
       51    60      3C   17D4          MOVZWL  (R0),R1                 ; Get the message length
  50      02 A0      DE   17D7          MOVAL   2(R0),R0                ; Point to the message text
52      0094'CF      DE   17DB          MOVAL   MASTER_NODE_DESC,R2
                          17E0          $FAO_S  CTRSTR = DEBUG_WRITE_MSG,- ; Form debugging message
                          17E0                  OUTLEN = DEBUG_PTR,-
                          17E0                  OUTBUF = DEBUG_FAO_BUF,-
                          17E0                  P1      = R1,-
                          17E0                  P2      = R0,-
                          17E0                  P3      = R2
           03AD      30   17F9          BSBW    GIVE_DEBUG_MSG          ; Let a debugging user see it
50      002C'CF      3C   17FC          MOVZWL  QUAD_STATUS,R0          ; Return $QIO result
                     04   1801          RET
```

```
                               1802  ;+
                               1802  ;       One of the DECnet read/write routines.  We have special conditions
                               1802  ;       here:  we are in our exit handler with System Service Failure mode
                               1802  ;       and ASTs turned off and we are reading the very file we we would
                               1802  ;       ordinarily be writing if we encounter an error.  We must therefore
                               1802  ;       make some assumptions and process errors locally.
                               1802  ;-
                               1802  SLAVE_EXIT_WRITE:
                         007C  1802          .WORD   ^M<R2,R3,R4,R5,R6>
                               1804
                               1804          $QIO_S  EFN    = #SS_SYNCH_EFN,- ; Copy a line of our error log file
                               1804                  CHAN   = NODE_CHANS,-
                               1804                  FUNC   = #IO$_WRITEVBLK,-
                               1804                  IOSB   = QUAD_STATUS,-
                               1804                  P1     = MESSAGE_BUFFER,-
                               1804                  P2     = #2*TEXTB_SIZE
                               1829          $SCHDWK_S DAYTIM = FIVE_SECONDS ; Allow a nominal time for the $QIO
                               183A          $HIBER_S                        ; Assume it will complete when we awaken
           002C'CF    B5       1841          TSTW    QUAD_STATUS             ; Did it complete though?
                05    12       1845          BNEQ    10$                     ; BR if it did
   002C'CF      01    B0       1847          MOVW    #1,QUAD_STATUS          ; Fool us into success - we can't wait
                               184C  10$:
                               184C          BLBSW   QUAD_STATUS,20$         ; BR if $QIO worked
   7E      002C'CF    3C       1854          MOVZWL  QUAD_STATUS,-(SP)       ; Otherwise...
   1BC3'CF      01    FB       1859          CALLS   #1,STATUS_TO_TEXT       ; ...set up...
       54    04 AC    D0       185E          MOVL    04(AP),R4               ; ...for an error message..
          53    64    3C       1862          MOVZWL  (R4),R3                 ; ...just as though...
       54    02 A4    DE       1865          MOVAL   2(R4),R4                ; ...we'd called...
   55      0094'CF    DE       1869          MOVAL   MASTER_NODE_DESC,R5     ; ...our regular error routines...
   56      00BB'CF    DE       186E          MOVAL   NULL,R6                 ; ...
                               1873          $FAO_S  CTRSTR = WRITE_MSG,-    ; ...
                               1873                  OUTLEN = BUFFER_PTR,-
                               1873                  OUTBUF = FAO_BUF,-
                               1873                  P1     = R3,-
                               1873                  P2     = R4,-
                               1873                  P3     = R5,-
                               1873                  P4     = R6
          56    5E    D0       188E          MOVL    SP,R6                   ; (This will clean up stack)
           0EDE'CF    DF       1891          PUSHAL  STATUS_PTR              ; ...
                01    DD       1895          PUSHL   #1
     00741132   8F    DD       1897          PUSHL   #UETP$_TEXT!STS$K_ERROR
           0CBC'CF    DF       189D          PUSHAL  BUFFER_PTR
     000F0001   8F    DD       18A1          PUSHL   #^XF0001
     00741132   8F    DD       18A7          PUSHL   #UETP$_TEXT!STS$K_ERROR
           0034'CF    D6       18AD          INCL    ERROR_COUNT
           0034'CF    DD       18B1          PUSHL   ERROR_COUNT
           0061'CF    DF       18B5          PUSHAL  NEWNAM_DESC
     00010002   8F    DD       18B9          PUSHL   #^X10002
     00748022   8F    DD       18BF          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR
                0A    DD       18C5          PUSHL   #10
          55    5E    D0       18C7          MOVL    SP,R5
                               18CA          $PUTMSG_S MSGVEC = (R5)         ; ...but use no AST and don't log it!
          5E    56    D0       18D9          MOVL    R6,SP                   ; Clean up the stack
                               18DC  20$:
       50    04 AC    D0       18DC          MOVL    04(AP),R0               ; Point to the message
          51    60    3C       18E0          MOVZWL  (R0),R1                 ; Get the message length
       50    02 A0    DE       18E3          MOVAL   2(R0),R0                ; Point to the message text
```

```
      52   0094'CF   DE  18E7  2691         MOVAL   MASTER_NODE_DESC,R2
                         18EC  2692         $FAO_S  CTRSTR = DEBUG_WRITE_MSG,- ; Form debugging message
                         18EC  2693                 OUTLEN = DEBUG_PTR,-
                         18EC  2694                 OUTBUF = DEBUG_FAO_BUF,-
                         18EC  2695                 P1     = R1,-
                         18EC  2696                 P2     = R0,-
                         18EC  2697                 P3     = R2
   11 0024'CF   00   E1  1905  2698         BBC     #CLIG_V_DEBUG,FLAGS,30$ ; Skip message if not debugging
                         190B  2699         $PUTMSG_S MSGVEC = DEBUG_QIO_MSG_PTR ; Print but don't log message!
                         191C  2700 30$:
      50   002C'CF   3C  191C  2701         MOVZWL  QUAD_STATUS,R0          ; Return $QIO result
                     04  1921  2702         RET
```

```
                               1922  ;+
                               1922  ;       One of the DECnet read/write routines.
                               1922  ;-
                               1922  MASTER_WRITE:
                       0000    1922          .WORD   ^M<>
                               1924
                               1924          $SETIMR_S DAYTIM = QIO_DELTA,-  ; Prevent hangs waiting for DECnet
                               1924                  ASTADR = TIME_OUT,-
                               1924                  REQIDT = AP
                               1937          $QIOW_S EFN    = #SS_SYNCH_EFN,-
                               1937                  CHAN   = 12(AP),-
                               1937                  FUNC   = #IO$_WRITEVBLK,-
                               1937                  IOSB   = QUAD_STATUS,-
                               1937                  P1     = MESSAGE_BUFFER,-
                               1937                  P2     = #TEXTB_SIZE
                               195B          $CANTIM_S REQIDT = AP           ; We returned from the DECnet QIO
        17 002C'CF      E8     1966          BLBS    QUAD_STATUS,10$         ; BR if message sent correctly
           0999'CF      DF     196B          PUSHAL  EXCLUDE_MSG             ; Complain if it was not
              08 AC     DD     196F          PUSHL   08(AP)
              04 AC     DD     1972          PUSHL   04(AP)
     1B38'CF    03      FB     1975          CALLS   #3,WRITE_FAILED
     50      08 AC      D0     197A          MOVL    08(AP),R0
     02 A0      02      A8     197E          BISW2   #CLIG_M_DEADNODE,2(R0)  ; We're done with this node
                               1982  10$:
     50      04 AC      D0     1982          MOVL    04(AP),R0               ; Point to the message
           51   60      3C     1986          MOVZWL  (R0),R1                 ; Get the message length
     50      02 A0      DE     1989          MOVAL   2(R0),R0                ; Point to the message text
                               198D          $FAO_S  CTRSTR = DEBUG_WRITE_MSG,- ; Form debug message
                               198D                  OUTLEN = DEBUG_PTR,-
                               198D                  OUTBUF = DEBUG_FAO_BUF,-
                               198D                  P1     = R1,-
                               198D                  P2     = R0,-
                               198D                  P3     = 08(AP)
              01FF      30     19A7          BSBW    GIVE_DEBUG_MSG          ; Let a debugging user see it
     50    002C'CF      3C     19AA          MOVZWL  QUAD_STATUS,R0          ; Return $QIO result
                        04     19AF          RET
```

```
                              19B0  ;+
                              19B0  ;       One of the DECnet read/write routines.
                              19B0  ;-
                              19B0  MASTER_READ:
                        0000  19B0          .WORD   ^M<>
                              19B2
                              19B2          $SETIMR_S DAYTIM = QIO_DELTA,-  ; Prevent hangs waiting for DECnet
                              19B2                  ASTADR = TIME_OUT,-
                              19B2                  REQIDT = AP
                              19C5          $QIOW_S EFN    = #SS_SYNCH_EFN,- ; See if other node acknowledges us
                              19C5                  CHAN   = 12(AP),-
                              19C5                  FUNC   = #IO$_READVBLK,-
                              19C5                  IOSB   = QUAD_STATUS,-
                              19C5                  P1     = BUFFER,-
                              19C5                  P2     = #TEXTB_SIZE
                              19E9          $CANTIM_S REQIDT = AP             ; We returned from the DECnet QIO
       17 002C'CF      E8     19F4          BLBS    QUAD_STATUS,10$           ; BR if message received correctly
          0999'CF      DF     19F9          PUSHAL  EXCLUDE_MSG               ; Complain if it was not
             08 AC     DD     19FD          PUSHL   08(AP)
             04 AC     DD     1A00          PUSHL   04(AP)
    1B29'CF    03      FB     1A03          CALLS   #3,READ_FAILED
       50    08 AC     D0     1A08          MOVL    08(AP),R0
       02 A0    02     A8     1A0C          BISW2   #CLIG_M_DEADNODE,2(R0)    ; We're done with this node
                              1A10  10$:
       50    04 AC     D0     1A10          MOVL    04(AP),R0                 ; Point to the message
          51    60     3C     1A14          MOVZWL  (R0),R1                   ; Get the message length
       50    02 A0     DE     1A17          MOVAL   2(R0),R0                  ; Point to the message text
                              1A1B          $FAO_S  CTRSTR = DEBUG_READ_MSG,- ; Form debug message
                              1A1B                  OUTLEN = DEBUG_PTR,-
                              1A1B                  OUTBUF = DEBUG_FAO_BUF,-
                              1A1B                  P1     = R1,-
                              1A1B                  P2     = R0,-
                              1A1B                  P3     = 08(AP)
             0171      30     1A35          BSBW    GIVE_DEBUG_MSG            ; Let debugging user see it
    50    002C'CF      3C     1A38          MOVZWL  QUAD_STATUS,R0            ; Return $QIO result
                       04     1A3D          RET
```

```
                        1A3E  ;+
                        1A3E  ;       One of the DECnet read/write routines.
                        1A3E  ;-
                        1A3E  MASTER_ERRORLOG_READ:
                  0000  1A3E          .WORD   ^M<>
                        1A40
                        1A40          $SETIMR_S DAYTIM = QIO_DELTA,-  ; Prevent hangs waiting for DECnet
                        1A40                    ASTADR = 100$,-
                        1A40                    REQIDT = AP
                        1A53          $QIOW_S EFN    = #SS_SYNCH_EFN,- ; See if other node acknowledges us
                        1A53                  CHAN   = 12(AP),-
                        1A53                  FUNC   = #IO$_READVBLK,-
                        1A53                  IOSB   = QUAD_STATUS,-
                        1A53                  P1     = BUFFER,-
                        1A53                  P2     = #2*TEXTB_SIZE
                        1A77          $CANTIM_S REQIDT = AP             ; We returned from the DECnet QIO
     0F 002C'CF    E8   1A82          BLBS    QUAD_STATUS,10$           ; BR if message received correctly
        09CD'CF    DF   1A87          PUSHAL  PLEASE_CHECK_MSG          ; Complain if it was not
          08 AC    DD   1A8B          PUSHL   08(AP)
          04 AC    DD   1A8E          PUSHL   04(AP)
  1B29'CF    03    FB   1A91          CALLS   #3,READ_FAILED
                        1A96  10$:
     50   04 AC    D0   1A96          MOVL    04(AP),R0                 ; Point to the message
        51    60   3C   1A9A          MOVZWL  (R0),R1                   ; Get the message length
     50   02 A0    DE   1A9D          MOVAL   2(R0),R0                  ; Point to the message text
                        1AA1          $FAO_S  CTRSTR = DEBUG_READ_MSG,- ; Form debugging message
                        1AA1                  OUTLEN = DEBUG_PTR,-
                        1AA1                  OUTBUF = DEBUG_FAO_BUF,-
                        1AA1                  P1     = R1,-
                        1AA1                  P2     = R0,-
                        1AA1                  P3     = 08(AP)
           00EB    30   1ABB          BSBW    GIVE_DEBUG_MSG            ; Let debugging user see it
   50   002C'CF    3C   1ABE          MOVZWL  QUAD_STATUS,R0            ; Return $QIO result
                   04   1AC3          RET
                        1AC4
                        1AC4
                        1AC4  100$:                                     ; Catch DECnet timeouts
                  0000  1AC4          .WORD   ^M<>
                        1AC6
     5C   04 AC    D0   1AC6          MOVL    04(AP),AP                 ; Get AP from DECnet read routine
     50   0C AC    3C   1ACA          MOVZWL  12(AP),R0                 ; Get the DECnet channel...
                        1ACE          $CANCEL_S CHAN = R0               ; ...because we can't wait forever
                   04   1AD8          RET
```

```
                            1AD9          .SBTTL  Timer Expiration Routine
                            1AD9  ;++
                            1AD9  ; FUNCTIONAL DESCRIPTION:
                            1AD9  ;         This routine will be called only if the timer goes off which was set to
                            1AD9  ;         prevent program hangs while waiting for the completion of a DECnet $QIO.
                            1AD9  ;
                            1AD9  ; CALLING SEQUENCE:
                            1AD9  ;         Called via AST at $SETIMR expiration.
                            1AD9  ;
                            1AD9  ; INPUT PARAMETERS:
                            1AD9  ;         04(AP)  Contents of AP when the $QIO was issued.  See ''Read and Write
                            1AD9  ;                 DECnet'' routines.
                            1AD9  ;
                            1AD9  ; IMPLICIT INPUTS:
                            1AD9  ;         NODE_CHANS has the DECnet channel (slave routines only)
                            1AD9  ;         Because we will use the AP from the DECnet read/write routines, we
                            1AD9  ;                 will have the DECnet channel for the master routines as 12(AP).
                            1AD9  ;
                            1AD9  ; OUTPUT PARAMETERS:
                            1AD9  ;         NONE
                            1AD9  ;
                            1AD9  ; IMPLICIT OUTPUTS:
                            1AD9  ;         NONE
                            1AD9  ;
                            1AD9  ; COMPLETION CODES:
                            1AD9  ;         NONE
                            1AD9  ;
                            1AD9  ; SIDE EFFECTS:
                            1AD9  ;         Message saying that the $QIO was cancelled.
                            1AD9  ;         QUAD_STATUS gets SS$_CANCEL or SS$_ABORT.
                            1AD9  ;
                            1AD9  ;--
                            1AD9
                            1AD9  TIME_OUT:
                      0004  1AD9          .WORD   ^M<R2>
                            1ADB
     5C     04 AC      D0   1ADB          MOVL    04(AP),AP               ; Get AP from DECnet read/write routine
  50    00AA'CF        3C   1ADF          MOVZWL  NODE_CHANS,R0           ; Get DECnet channel assuming a slave
  52    0094'CF        DE   1AE4          MOVAL   MASTER_NODE_DESC,R2     ; Get node name assuming a slave
        6C     01      D1   1AE9          CMPL    #1,00(AP)               ; But was it?  Slaves have only 1 arg
               08      13   1AEC          BEQL    10$                     ; BR if so - we're set up
     50     0C AC      3C   1AEE          MOVZWL  12(AP),R0               ; It was master - get DECnet channel...
     52     08 AC      D0   1AF2          MOVL    08(AP),R2               ; ...and node name
                            1AF6  10$:
                            1AF6          $CANCEL_S CHAN = R0             ; We can't wait forever for DECnet
                            1B00          $FAO_S  CTRSTR = CANCEL_MSG,-   ; Let the user know what happened
                            1B00                  OUTLEN = BUFFER_PTR,-
                            1B00                  OUTBUF = FAO_BUF,-
                            1B00                  P1     = R2
                            1B15          $PUTMSG_S MSGVEC = CANCEL_MSG_PTR,-
                            1B15                  ACTRTN = SE_COPY
                        04  1B28          RET
```

```
                           1B29         .SBTTL  Form DECnet Error Messages
                           1B29  ;++
                           1B29  ; FUNCTIONAL DESCRIPTION:
                           1B29  ;         A set of common routines to format and issue typical error messages
                           1B29  ;         from reading or writing to DECnet.
                           1B29  ;
                           1B29  ; CALLING SEQUENCE:
                           1B29  ;         CALLS   #3,READ_FAILED or WRITE_FAILED or GARBLED_TRANS
                           1B29  ;
                           1B29  ; INPUT PARAMETERS:
                           1B29  ;         12(AP)  address of .ASCID giving consequence of error
                           1B29  ;         08(AP)  address of .ASCID node name from which error occurred
                           1B29  ;         04(AP)  MESSAGE_NAMES message name (count word followed by text)
                           1B29  ;
                           1B29  ; IMPLICIT INPUTS:
                           1B29  ;         QUAD_STATUS has failure code if this was called after a $QIO
                           1B29  ;
                           1B29  ; OUTPUT PARAMETERS:
                           1B29  ;         NONE
                           1B29  ;
                           1B29  ; IMPLICIT OUTPUTS:
                           1B29  ;         NONE
                           1B29  ;
                           1B29  ; COMPLETION CODES:
                           1B29  ;         NONE (R0 is garbage)
                           1B29  ;
                           1B29  ; SIDE EFFECTS:
                           1B29  ;         Error message signalled.
                           1B29  ;         STATUS_PTR, STATUS_BUFFER, BUFFER_PTR, BUFFER written over.
                           1B29  ;--
                           1B29
                           1B29  READ_FAILED:
                   003C    1B29          .WORD   ^M<R2,R3,R4,R5>
                           1B2B
 55    08E0'CF       7E    1B2B          MOVAQ   READ_MSG,R5             ; Get the address of the message
           27        10    1B30          BSBB    COMMON_MSG              ; Join common code
 1DAD'CF   06        FB    1B32          CALLS   #6,ERROR_SIGNAL         ; Signal the error
                     04    1B37          RET
                           1B38
                           1B38  WRITE_FAILED:
                   003C    1B38          .WORD   ^M<R2,R3,R4,R5>
                           1B3A
 55    08A9'CF       7E    1B3A          MOVAQ   WRITE_MSG,R5            ; Get the address of the message
           18        10    1B3F          BSBB    COMMON_MSG              ; Join common code
 1DAD'CF   06        FB    1B41          CALLS   #6,ERROR_SIGNAL         ; Signal the error
                     04    1B46          RET
                           1B47
                           1B47  GARBLED_TRANS:
                   003C    1B47          .WORD   ^M<R2,R3,R4,R5>
                           1B49
 55    0918'CF       7E    1B49          MOVAQ   GARBLE_MSG,R5           ; Get the address of the message
           09        10    1B4E          BSBB    COMMON_MSG              ; Join common code
 1DAD'CF   03        FB    1B50          CALLS   #3,ERROR_SIGNAL         ; Signal the error
      5E   0C        C0    1B55          ADDL2   #12,SP                  ; Get rid of extra COMMON_MSG args
                     04    1B58          RET
```

```
                        1B59  COMMON_MSG:
             04   BA    1B59        POPR     #^M<R2>                     ; Get return PC
  7E   002C'CF    3C    1B5B        MOVZWL   QUAD_STATUS,-(SP)           ; Set up $QIO status if msg needs it
 1BC3'CF    01    FB    1B60        CALLS    #1,STATUS_TO_TEXT           ; Get message text for that status
  54    04 AC     D0    1B65        MOVL     04(AP),R4                   ; Point to MESSAGE_NAMES length
       53   64    3C    1B69        MOVZWL   (R4),R3                     ; Get the length of message type
  54    02 A4     DE    1B6C        MOVAL    2(R4),R4                    ; Point to the text naming the message
                        1B70        $FAO_S   CTRSTR = (R5),-             ; Form the message text
                        1B70                 OUTLEN = BUFFER_PTR,-
                        1B70                 OUTBUF = FAO_BUF,-
                        1B70                 P1     = R3,-
                        1B70                 P2     = R4,-
                        1B70                 P3     = 08(AP),-
                        1B70                 P4     = 12(AP)
        0EDE'CF   DF    1B8B        PUSHAL   STATUS_PTR                  ; Set up SIGNAL info for $QIO status
             01   DD    1B8F        PUSHL    #1
    00741132 8F   DD    1B91        PUSHL    #UETP$_TEXT!STS$K_ERROR
        0CBC'CF   DF    1B97        PUSHAL   BUFFER_PTR                  ; Set up rest of SIGNAL info
    000F0001 8F   DD    1B9B        PUSHL    #^XF0001
    00741132 8F   DD    1BA1        PUSHL    #UETP$_TEXT!STS$K_ERROR
             62   17    1BA7        JMP      (R2)                        ; Subroutine return
```

```
                              1BA9        .SBTTL  Tracing Messages Routine
                              1BA9   ;++
                              1BA9   ; FUNCTIONAL DESCRIPTION:
                              1BA9   ;       Outputs a trace message for debugging purposes, if appropriate.
                              1BA9   ;
                              1BA9   ; IMPLICIT INPUTS:
                              1BA9   ;       DEBUG_PTR is a descriptor for the message.
                              1BA9   ;       FLAGS has a switch to indicate debugging mode
                              1BA9   ;
                              1BA9   ; IMPLICIT OUTPUTS:
                              1BA9   ;       NONE
                              1BA9   ;
                              1BA9   ; SIDE EFFECTS:
                              1BA9   ;       Message to SYS$OUTPUT/SYS$ERROR if we are in debugging mode
                              1BA9   ;       Message copied to slave's SYS$ERROR.LOG, if appropriate
                              1BA9   ;
                              1BA9   ;--
                              1BA9
                              1BA9   GIVE_DEBUG_MSG:
    13 0024'CF   00    E1     1BA9           BBC     #CLIG_V_DEBUG,FLAGS,10$ ; Skip message if not tracing
                              1BAF           $PUTMSG_S MSGVEC = DEBUG_QIO_MSG_PTR,-
                              1BAF                   ACTRTN = SE_COPY
                              1BC2   10$:
                       05     1BC2           RSB
```

```
UET
Sym

$$.
$$.
$$.
$$.
$$.
$$.
$$.
$$T
$$T
ABO
ACC
ACC
ANN
ARG
BLA
BLA
BLO
BRK
BRK
BRK
BRK
BUF
BUF
CAN
CAN
CCA
CHE
CHE
CHF
CHF
CHF
CHF
CLI
CLI
CLI
CLI
CLI
CLI
CLI
CLI
CLI
CLI
CLI
CLS
CLS
CLS
CLS
CLU
CLU
COM
COM
CON
CON
CRL
CUR
CUR
DC$
```

```
                              1BC3          .SBTTL  STATUS_TO_TEXT - Get Text Associated with a Status Value
                              1BC3   ;++
                              1BC3   ; FUNCTIONAL DESCRIPTION:
                              1BC3   ;       To enable more useful error messages, we'd like to print out the
                              1BC3   ;       message associated with failures as well as the messages we provide
                              1BC3   ;       ourself.  Some of the messages have $FAO arguments, the values
                              1BC3   ;       for which are lost.  Provide the fac-s-abbrev,text for each message,
                              1BC3   ;       but with the $FAO directives intact.
                              1BC3   ;
                              1BC3   ; CALLING SEQUENCE:
                              1BC3   ;       PUSHL   status
                              1BC3   ;       CALLS   #1,STATUS_TO_TEXT
                              1BC3   ;
                              1BC3   ; INPUT PARAMETERS:
                              1BC3   ;       04(AP)  VMS status (message number and severity)
                              1BC3   ;
                              1BC3   ; IMPLICIT INPUTS:
                              1BC3   ;       STATUS_STRING has an introductory message
                              1BC3   ;
                              1BC3   ; OUTPUT PARAMETERS:
                              1BC3   ;       NONE
                              1BC3   ;
                              1BC3   ; IMPLICIT OUTPUTS:
                              1BC3   ;       STATUS_PTR has a descriptor for our message in STATUS_BUFFER
                              1BC3   ;
                              1BC3   ; COMPLETION CODES:
                              1BC3   ;       Status from $GETMSG
                              1BC3   ;
                              1BC3   ; SIDE EFFECTS:
                              1BC3   ;       NONE
                              1BC3   ;--
                              1BC3
                              1BC3   STATUS_TO_TEXT:
                        00FC  1BC3          .WORD   ^M<R2,R3,R4,R5,R6,R7>   ; Entry mask
                              1BC5
 0EDE'CF    010D 8F       3C  1BC5          MOVZWL  #TEXTB_SIZE,STATUS_PTR  ; Set the size of our return buffer
                              1BCC          $GETMSG_S MSGID = 04(AP),-      ; Get the message
                              1BCC                  MSGLEN = STATUS_PTR,-
                              1BCC                  BUFADR = STATUS_PTR
                 01       BB  1BE2          PUSHR   #^M<R0>                 ; Save this as final status
       56   0158'CF       3C  1BE4          MOVZWL  STATUS_STRING,R6        ; Get the length of our intro text
       57   0EE6'CF       DE  1BE9          MOVAL   STATUS_BUFFER,R7        ; Point to just beyond where...
            57   56       C0  1BEE          ADDL2   R6,R7                   ; ...the intro would end in our buffer
            0EDE'CF       28  1BF1          MOVC3   STATUS_PTR,-            ; Shift the message...
       67   0EE6'CF           1BF5                  STATUS_BUFFER,(R7)      ; ...by the length of the intro...
            57   53       D0  1BF9          MOVL    R3,R7
    0160'CF      56       28  1BFC          MOVC3   R6,STATUS_STRING+8,-    ; ...so we may surround message...
            0EE6'CF           1C01                  STATUS_BUFFER
            87   22       90  1C04          MOVB    #^A/"/,(R7)+            ; ...with our intro
       56   0EE6'CF       DE  1C07          MOVAL   STATUS_BUFFER,R6        ; Get the length...
 0EDE'CF    57   56       C3  1C0C          SUBL3   R6,R7,STATUS_PTR        ; ...of the entire mess
                 01       BA  1C12          POPR    #^M<R0>                 ; Restore $GETMSG status
                 04       1C14          RET
```

```
UET
Sym

FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAO
FIL
FIL
FIV
FLA
GAR
GAR
GET
GET
GIV
GOT
GOT
HEL
HEL
IMO
IMO
IND
INP
IO$
IO$
IO$
IO$
JPI
LCK
LCK
LCK
LCK
LIB
LIN
LON
LON
MAS
MAS
MAS
MAS
MAS
MAS
MAX
MAX
MEM
MEM
MES
MES
MOD
MOV
```

```
                          1C15          .SBTTL  System Service Exception Handler
                          1C15   ;++
                          1C15   ; FUNCTIONAL DESCRIPTION:
                          1C15   ;      This routine is executed if a software or hardware exception occurs or
                          1C15   ;      if a LIB$SIGNAL system service is used to output a message.
                          1C15   ;
                          1C15   ; CALLING SEQUENCE:
                          1C15   ;      Entered via an exception from the system
                          1C15   ;
                          1C15   ; INPUT PARAMETERS:
                          1C15   ;      Signal and mechanism arrays from an exception vector
                          1C15   ;
                          1C15   ; IMPLICIT INPUTS:
                          1C15   ;      ERROR_COUNT has the previous cumulative error count
                          1C15   ;
                          1C15   ; OUTPUT PARAMETERS:
                          1C15   ;      NONE
                          1C15   ;
                          1C15   ; IMPLICIT OUTPUTS:
                          1C15   ;      EXIT_STATUS contains error code if we exit
                          1C15   ;
                          1C15   ; COMPLETION CODES:
                          1C15   ;      SS$_NORMAL if it's a UETP condition or RMS error.
                          1C15   ;      Error status from exception, otherwise.
                          1C15   ;
                          1C15   ; SIDE EFFECTS:
                          1C15   ;      STATUS_PTR, STATUS_BUFFER get used.
                          1C15   ;      May branch to ERROR_EXIT.
                          1C15   ;      May print a message.
                          1C15   ;--
                          1C15
                          1C15   SSERROR:
                  0FFC    1C15          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                          1C17
                          1C17          $SETAST_S ENBFLG = #0           ; Disable AST delivery
               01   DD    1C20          PUSHL   #1                      ; Assume ASTs were enabled
         50    00'  D1    1C22          CMPL    S^#SS$_WASSET,R0        ; Were ASTs enabled?
               02   13    1C25          BEQL    10$                     ; BR if they were
               6E   D4    1C27          CLRL    (SP)                    ; Set ASTs to remain disabled
                          1C29   10$:
                          1C29          $SETSFM_S ENBFLG = #0           ; Disable SS failure mode
               01   DD    1C32          PUSHL   #1                      ; Assume SS failure mode was enabled
         50    00'  D1    1C34          CMPL    S^#SS$_WASSET,R0        ; Was SS failure mode enabled?
               02   13    1C37          BEQL    20$                     ; BR if it was
               6E   D4    1C39          CLRL    (SP)                    ; Set SS failure mode to remain off
                          1C3B   20$:
      56    04 AC   D0    1C3B          MOVL    CHF$L_SIGARGLST(AP),R6  ; Get the signal array pointer
      59    04 A6   7D    1C3F          MOVQ    CHF$L_SIG_NAME(R6),R9   ; Get NAME in R9 and ARG1 in R10
               10   ED    1C43          CMPZV   #STS$V_FAC_NO,-         ; Is this a message from LIB$SIGNAL?
               0C         1C45                  #STS$S_FAC_NO,-
 00000074 8F   59         1C46                  R9,#UETP$_FACILITY
               16   12    1C4C          BNEQ    30$                     ; BR if this is not a UETP exception
         66    02   C2    1C4E          SUBL2   #2,CHF$L_SIG_ARGS(R6)   ; Drop the PC and PSL
                          1C51          $PUTMSG_S MSGVEC = -            ; Print the message
                          1C51                  CHF$L_SIG_ARGS(R6),-
                          1C51                ACTRTN = SE_COPY
               21   11    1C62          BRB     40$                     ; Restore ASTs and SS fail mode
```

```
                                   1C64  3089 30$:
    59      00000000'8F   D1  1C64  3090          CMPL    #SS$_SSFAIL,R9           ; RMS failures are SysSvc failures
                     32   12  1C6B  3091          BNEQ    50$                      ; BR if this can't be an RMS failure
                     10   ED  1C6D  3092          CMPZV   #STS$V_FAC_NO,-          ; Is it an RMS failure?
                     0C       1C6F  3093                  #STS$S_FAC_NO,-
              01     5A       1C70  3094                  R10,#RMS$_FACILITY
                     2B   12  1C72  3095          BNEQ    50$                      ; BR if not
    5A      F0000000 8F   CA  1C74  3096          BICL2   #^XF0000000,R10          ; Strip control bits from status code
            08 A6    04   39  1C7B  3097          MATCHC  #4,CHF$L_SIG_ARG1(R6),-  ; Is it an RMS failure for which...
                     14       1C7F  3098                  #NRAT_LENGTH,-
              0D9E'CF         1C80  3099                  NO_RMS_AST_TABLE         ; ...no AST can be delivered?
                     1A   13  1C83  3100          BEQL    50$                      ; BR if so - must give error here
                              1C85  3101 40$:
              01     BA       1C85  3102          POPR    #^M<R0>                  ; Restore SS failure mode...
                              1C87  3103          $SETSFM_S ENBFLG = R0            ; ...
              01     BA       1C90  3104          POPR    #^M<R0>                  ; Restore AST enable...
                              1C92  3105          $SETAST_S ENBFLG = R0            ; ...
              50     00'  D0  1C9B  3106          MOVL    S^#SS$_NORMAL,R0         ; Supply a standard status for exit
                     04       1C9E  3107          RET                              ; Resume processing (or goto RMS_ERROR)
                              1C9F  3108 50$:
         0028'CF     59   D0  1C9F  3109          MOVL    R9,EXIT_STATUS           ; Save the status
                     58   D4  1CA4  3110          CLRL    R8                       ; Assume for now it's not SS failure
0028'CF     00000000'8F   D1  1CA6  3111          CMPL    #SS$_SSFAIL,EXIT_STATUS  ; But is it a System Service failure?
                     1C   12  1CAF  3112          BNEQ    60$                      ; BR if not - no special case message
                     5A   DD  1CB1  3113          PUSHL   R10                      ; Get the text...
         FF0B CF     01   FB  1CB3  3114          CALLS   #1,STATUS_TO_TEXT        ; ...associated with this specific error
              0EDE'CF     DF  1CB8  3115          PUSHAL  STATUS_PTR               ; Build up a message describing...
                     01   DD  1CBC  3116          PUSHL   #1                       ; ...why the System Service failed
                     00   EF  1CBE  3117          EXTZV   #STS$V_SEVERITY,-        ; Give the message...
          7E     5A  03       1CC0  3118                  #STS$S_SEVERITY,R10,-(SP) ; ...the correct severity code,....
    6E      00741130 8F   C8  1CC3  3119          BISL2   #UETP$_TEXT,(SP)         ; ...facility and id
              58     03   D0  1CCA  3120          MOVL    #3,R8                    ; Count the number of args we pushed
                              1CCD  3121 60$:
         57     66   04   C5  1CCD  3122          MULL3   #4,CHF$L_SIG_ARGS(R6),R7 ; Get arglist length in bytes
                5E   57   C2  1CD1  3123          SUBL2   R7,SP                    ; Save the current signal array...
    6E       04 A6   57   28  1CD4  3124          MOVC3   R7,CHF$L_SIG_NAME(R6),(SP) ; ...on the stack
         7E     66   58   C1  1CD9  3125          ADDL3   R8,CHF$L_SIG_ARGS(R6),-(SP) ; Push the current arg count
                   0120   31  1CDD  3126          BRW     ERROR_EXIT
```

```
                           1CE0         .SBTTL  Action Routine for Slave's SYS$ERROR.LOG
                           1CE0   ;++
                           1CE0   ; FUNCTIONAL DESCRIPTION:
                           1CE0   ;       This routine decides if a message is to be written to SYS$ERROR.LOG
                           1CE0   ;       (a slave's copy of its SYS$ERROR which will be relayed to the master
                           1CE0   ;       process at the end of testing) and writes it there if appropriate.
                           1CE0   ;
                           1CE0   ; CALLING SEQUENCE:
                           1CE0   ;       Called as a $PUTMSG action routine.
                           1CE0   ;
                           1CE0   ; INPUT PARAMETERS:
                           1CE0   ;       04(AP)  Address of a string descriptor for the message $PUTMSG
                           1CE0   ;               intends to write
                           1CE0   ;
                           1CE0   ; IMPLICIT INPUTS
                           1CE0   ;       FLAGS(CLIG_M_SLAVE) is on iff we're a slave process.
                           1CE0   ;
                           1CE0   ; OUTPUT PARAMETERS:
                           1CE0   ;       NONE
                           1CE0   ;
                           1CE0   ; IMPLICIT OUTPUTS:
                           1CE0   ;       NONE
                           1CE0   ;
                           1CE0   ; COMPLETION CODES:
                           1CE0   ;       R0 contains an odd number so $PUTMSG may write its message
                           1CE0   ;
                           1CE0   ; SIDE EFFECTS:
                           1CE0   ;       Slave's SYS$ERROR.LOG written if appropriate
                           1CE0   ;--
                           1CE0
                           1CE0   SE_COPY:
                     0000  1CE0         .WORD   ^M<>
                           1CE2
24 0024'CF   01   E1  1CE2         BBC     #CLIG_V_SLAVE,FLAGS,10$ ; Skip this if we're the master node
1E 0024'CF   02   E0  1CE8         BBS     #CLIG_V_SE_DEAD,FLAGS,10$ ; Also skip if we can't write to log
     50   04 AC   D0  1CEE         MOVL    04(AP),R0               ; Point to the message buffer desc
   1502'CF   60   B0  1CF2         MOVW    (R0),SE_RAB+RAB$W_RSZ   ; Set up the message size...
1508'CF   04 A0   D0  1CF7         MOVL    4(R0),SE_RAB+RAB$L_RBF  ; ...and address
                      1CFD         $PUT    RAB = SE_RAB,-          ; Write the message
                      1CFD                 ERR = RMS_ERROR
                      1D0C   10$:
          50   01  D0  1D0C         MOVL    #1,R0                   ; Supply an exit status for $PUTMSG
                   04  1D0F         RET
```

```
                          1D10        .SBTTL  RMS Error Handler
                          1D10  ;++
                          1D10  ; FUNCTIONAL DESCRIPTION:
                          1D10  ;       This routine handles error returns from RMS calls.
                          1D10  ;
                          1D10  ; CALLING SEQUENCE:
                          1D10  ;       Called by RMS when a file processing error is found.
                          1D10  ;
                          1D10  ; INPUT PARAMETERS:
                          1D10  ;       The FAB or RAB associated with the RMS call.
                          1D10  ;
                          1D10  ; IMPLICIT INPUTS:
                          1D10  ;       NONE
                          1D10  ;
                          1D10  ; OUTPUT PARAMETERS:
                          1D10  ;       NONE
                          1D10  ;
                          1D10  ; IMPLICIT OUTPUTS:
                          1D10  ;       NONE
                          1D10  ;
                          1D10  ; COMPLETION CODES:
                          1D10  ;       NONE
                          1D10  ;
                          1D10  ; SIDE EFFECTS:
                          1D10  ;       Error message
                          1D10  ;
                          1D10  ;--
                          1D10
                          1D10  RMS_ERROR:
                   0FFC   1D10          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                          1D12
  56    04 AC      D0     1D12          MOVL    4(AP),R6                ; See whether we're dealing with...
     66    03      91     1D16          CMPB    #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
           10      12     1D19          BNEQ    10$                     ; BR if it's a RAB
  57   011D'CF     DE     1D1B          MOVAL   FILE,R7                 ; FAB-specific code:  text string...
     58    56      D0     1D20          MOVL    R6,R8                   ; ...address of FAB...
        0C A6      DD     1D23          PUSHL   FAB$L_STV(R6)           ; ...STV field for error...
        08 A6      DD     1D26          PUSHL   FAB$L_STS(R6)           ; ...and STS field for error
           0F      11     1D29          BRB     20$                     ; FAB and RAB share other code
                          1D2B  10$:
  57   0129'CF     DE     1D2B          MOVAL   RECORD,R7               ; RAB-specific code:  text string...
   58   3C A6      D0     1D30          MOVL    RAB$L_FAB(R6),R8        ; ...address of associated FAB...
        0C A6      DD     1D34          PUSHL   RAB$L_STV(R6)           ; ...STV field for error...
        08 A6      DD     1D37          PUSHL   RAB$L_STS(R6)           ; ...and STS field for error
                          1D3A  20$:
  50   1430'CF     DE     1D3A          MOVAL   SE_FAB,R0               ; Check to see...
     58    50      D1     1D3F          CMPL    R0,R8                   ; ...if the error was in SYS$ERROR.LOG
           05      12     1D42          BNEQ    30$                     ; BR if it was not
0024'CF    04      C8     1D44          BISL2   #CLIG_M_SE_DEAD,FLAGS   ; Prevent endless loop trying to log it
                          1D49  30$:
  5A    34 A8      9A     1D49          MOVZBL  FAB$B_FNS(R8),R10       ; Get the file name size
                          1D4D          $FAO_S  CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error msg...
                          1D4D                  OUTLEN = BUFFER_PTR,-
                          1D4D                  OUTBUF = FAO_BUF,-
                          1D4D                  P1     = R7,-
                          1D4D                  P2     = R10,-
                          1D4D                  P3     = FAB$L_FNA(R8)
```

```
          0CBC'CF   DF  1D67  3229          PUSHAL  BUFFER_PTR                  ; ...
       000F0001 8F  DD  1D6B  3230          PUSHL   #^XF0001                    ; ...
       00741132 8F  DD  1D71  3231          PUSHL   #UETP$_TEXT!STS$K_ERROR     ; ...and arguments for ERROR_SIGNAL
       1DAD'CF   05 FB  1D77  3232          CALLS   #5,ERROR_SIGNAL             ; Give the message
                    04  1D7C  3233          RET
```

```
                             1D7D          .SBTTL  CTRL/C Handler
                             1D7D  ;++
                             1D7D  ; FUNCTIONAL DESCRIPTION:
                             1D7D  ;       This routine handles CTRL/C AST's
                             1D7D  ;
                             1D7D  ; CALLING SEQUENCE:
                             1D7D  ;       Called via AST
                             1D7D  ;
                             1D7D  ; INPUT PARAMETERS:
                             1D7D  ;       NONE
                             1D7D  ;
                             1D7D  ; IMPLICIT INPUTS:
                             1D7D  ;       NONE
                             1D7D  ;
                             1D7D  ; OUTPUT PARAMETERS:
                             1D7D  ;       NONE
                             1D7D  ;
                             1D7D  ; IMPLICIT OUTPUTS:
                             1D7D  ;       NONE
                             1D7D  ;
                             1D7D  ; COMPLETION CODES:
                             1D7D  ;       SS$_CONTROLC with warning status
                             1D7D  ;
                             1D7D  ; SIDE EFFECTS:
                             1D7D  ;       Control-C message is signalled.
                             1D7D  ;       Program exits.
                             1D7D  ;
                             1D7D  ;--
                             1D7D
                             1D7D  CCASTHAND:
                      0FFC   1D7D          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                             1D7F
         7E   0000'8F   3C   1D7F          MOVZWL  #SS$_CONTROLC,-(SP)
                    00  DD   1D84          PUSHL   #0                      ; Indicate an abnormal termination
               0000'CF  DF   1D86          PUSHAL  PROCESS_NAME            ; ...
                    02  DD   1D8A          PUSHL   #2                      ; ...
           007410E0 8F  DD   1D8C          PUSHL   #UETP$_ABENDD!STS$K_WARNING ; ...
       00000000'GF  05  FB   1D92          CALLS   #5,G^LIB$SIGNAL         ; Output the message
                        D0   1D99          MOVL    #<STS$M_INHIB_MSG!-     ; Set the exit status
                             1D9A                  SS$_CONTROLC-=
                             1D9A                  STS$K_SUCCESS+STS$K_WARNING>,-
 0028'CF   0FFFFFFF'8F       1D9A                  EXIT_STATUS
                             1DA2          $EXIT_S CODE = EXIT_STATUS      ; Terminate program cleanly
```

```
                              1DAD          .SBTTL  ERROR_SIGNAL
                              1DAD  ;++
                              1DAD  ; FUNCTIONAL DESCRIPTION:
                              1DAD  ;       This routine prints an error message with the standard UETP error box.
                              1DAD  ;
                              1DAD  ; CALLING SEQUENCE:
                              1DAD  ;       PUSHL   arguments to LIB$SIGNAL
                              1DAD  ;       CALLS   count of above,ERROR_SIGNAL
                              1DAD  ;
                              1DAD  ; INPUT PARAMETERS:
                              1DAD  ;       Arguments to LIB$SIGNAL, as above
                              1DAD  ;
                              1DAD  ; IMPLICIT INPUTS:
                              1DAD  ;       ERROR_COUNT has a cumulative count of errors we've seen
                              1DAD  ;
                              1DAD  ; OUTPUT PARAMETERS:
                              1DAD  ;       NONE
                              1DAD  ;
                              1DAD  ; IMPLICIT OUTPUTS:
                              1DAD  ;       ERROR_COUNT is incremented
                              1DAD  ;
                              1DAD  ; COMPLETION CODES:
                              1DAD  ;       NONE
                              1DAD  ;
                              1DAD  ; SIDE EFFECTS:
                              1DAD  ;       Message to SYS$OUTPUT and SYS$ERROR
                              1DAD  ;
                              1DAD  ;--
                              1DAD
                              1DAD  ERROR_SIGNAL:
                       003C   1DAD          .WORD   ^M<R2,R3,R4,R5>
                              1DAF
                              1DAF          $SETAST_S ENBFLG = #0           ; ASTs can play havoc with messages
                  01     DD   1DB8          PUSHL   #1                      ; Assume ASTs were enabled
           50     00'    B1   1DBA          CMPW    S^#SS$_WASSET,R0        ; Were ASTs enabled?
                  02     13   1DBD          BEQL    10$                     ; BR if they were
                  6E     D4   1DBF          CLRL    (SP)                    ; Set ASTs to remain disabled
                              1DC1  10$:
 0038'CF    04    6C     C1   1DC1          ADDL3   00(AP),#4,ARG_COUNT     ; Get total number of args
      50    04    6C     C5   1DC7          MULL3   00(AP),#4,R0            ; Figure its length in bytes...
                  5E     50   C2   1DCB     SUBL2   R0,SP                   ; ...so we can...
    6E    04 AC   50     28   1DCE          MOVC3   R0,04(AP),(SP)          ; ...set up a list for LIB$SIGNAL
             0034'CF     D6   1DD3          INCL    ERROR_COUNT             ; Keep running error count
             0034'CF     DD   1DD7          PUSHL   ERROR_COUNT             ; Finish off arg list...
             0061'CF     DF   1DDB          PUSHAL  NEWNAM_DESC             ; ...
         00010002 8F     DD   1DDF          PUSHL   #^X10002                ; ...
         00748022 8F     DD   1DE5          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
00000000'GF  0038'CF     FB   1DEB          CALLS   ARG_COUNT,G^LIB$SIGNAL  ; Truly bitch
                  01     BA   1DF4          POPR    #^M<R0>                 ; Restore AST enable...
                              1DF6          $SETAST_S ENBFLG = R0           ; ...to its previous situation
                         04   1DFF          RET
```

```
                                1E00         .SBTTL  Error Exit
                                1E00   ;++
                                1E00   ; FUNCTIONAL DESCRIPTION:
                                1E00   ;       This routine prints an error message and exits.
                                1E00   ;
                                1E00   ; CALLING SEQUENCE:
                                1E00   ;       MOVx  error status value,EXIT_STATUS
                                1E00   ;       PUSHx error specific information on the stack
                                1E00   ;       PUSHL current argument count
                                1E00   ;       BRW   ERROR_EXIT
                                1E00   ;
                                1E00   ; INPUT PARAMETERS:
                                1E00   ;       Arguments to LIB$SIGNAL, as above
                                1E00   ;
                                1E00   ; IMPLICIT INPUTS:
                                1E00   ;       ERROR_COUNT has a cumulative count of errors we've seen
                                1E00   ;
                                1E00   ; OUTPUT PARAMETERS:
                                1E00   ;       Message to SYS$OUTPUT and SYS$ERROR
                                1E00   ;
                                1E00   ; IMPLICIT OUTPUTS:
                                1E00   ;       ERROR_COUNT is incremented
                                1E00   ;
                                1E00   ; COMPLETION CODES:
                                1E00   ;       UETP$_ABENDD with error status as a default
                                1E00   ;
                                1E00   ; SIDE EFFECTS:
                                1E00   ;       Program exits
                                1E00   ;
                                1E00   ;--
                                1E00
                                1E00   ERROR_EXIT:
                                1E00
                                1E00           $SETAST_S ENBFLG = #0           ; ASTs can play havoc with messages
      13 0024'CF    03    E0    1E09           BBS     #CLIG_V_BEGINMSG,FLAGS,10$ ; BR if "begin" msg already given
                                1E0F           $PUTMSG_S MSGVEC = CLIG_ANNOUNCE,- ; Give a beginning message if not
                                1E0F                   ACTRTN = SE_COPY
                                1E22   10$:
 0038'CF    08    8E    C1      1E22           ADDL3   (SP)+,#8,ARG_COUNT      ; Get total # args, pop partial count
            0034'CF     D6      1E28           INCL    ERROR_COUNT             ; Keep running error count
                 00     DD      1E2C           PUSHL   #0                      ; Push the time parameter
            0000'CF     DF      1E2E           PUSHAL  PROCESS_NAME            ; Push test name...
        000F0002 8F     DD      1E32           PUSHL   #^XF0002                ; ...arg count...
        007410E2 8F     DD      1E38           PUSHL   #UETP$_ABENDD!STS$K_ERROR ; ...and signal name
            0034'CF     DD      1E3E           PUSHL   ERROR_COUNT             ; Finish off arg list...
            0061'CF     DF      1E42           PUSHAL  NEWNAM_DESC             ; ...
        00010002 8F     DD      1E46           PUSHL   #^X10002                ; ...
        00748022 8F     DD      1E4C           PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR
            0038'CF     DD      1E52           PUSHL   ARG_COUNT               ; ...for error box message
            52    5E    D0      1E56           MOVL    SP,R2                   ; Keep a pointer to the MSGVEC
                                1E59           $PUTMSG_S MSGVEC = (R2),-       ; Truly bitch
                                1E59                   ACTRTN = SE_COPY
                                1E6A
            0028'CF     D5      1E6A           TSTL    EXIT_STATUS             ; Did we exit with an error code?
                 09     12      1E6E           BNEQ    20$                     ; BR if we did
        007410E2 8F     D0      1E70           MOVL    #UETP$_ABENDD!STS$K_ERROR,- ; Supply a generic one otherwise
            0028'CF             1E76                   EXIT_STATUS
```

```
                              1E79  3388 20$:
          10000000 8F    C8   1E79  3389         BISL    #STS$M_INHIB_MSG,-        ; Don't print messages twice!
              0028'CF         1E7F  3390                 EXIT_STATUS
                              1E82  3391         $EXIT_S CODE = EXIT_STATUS        ; Exit in error
```

```
                                1E8D           .SBTTL  Exit Handler
                                1E8D  ;++
                                1E8D  ; FUNCTIONAL DESCRIPTION:
                                1E8D  ;        This routine handles cleanup at exit.  For slave processes, it also
                                1E8D  ;        copies SYS$ERROR.LOG file to the master process.
                                1E8D  ;
                                1E8D  ; CALLING SEQUENCE:
                                1E8D  ;        Invoked automatically by $EXIT System Service.
                                1E8D  ;
                                1E8D  ; INPUT PARAMETERS:
                                1E8D  ;        EXIT_STATUS  contains the exit status.
                                1E8D  ;
                                1E8D  ; IMPLICIT INPUTS:
                                1E8D  ;        SYS$ERROR.LOG contains all slave messages that have gone to SYS$ERROR
                                1E8D  ;
                                1E8D  ; OUTPUT PARAMETERS:
                                1E8D  ;        NONE
                                1E8D  ;
                                1E8D  ; IMPLICIT OUTPUTS:
                                1E8D  ;        NONE
                                1E8D  ;
                                1E8D  ; COMPLETION CODES:
                                1E8D  ;        NONE
                                1E8D  ;
                                1E8D  ; SIDE EFFECTS:
                                1E8D  ;        Message announcing the end of the test.
                                1E8D  ;        For slave processes, SYS$ERROR.LOG gets copied to the master.
                                1E8D  ;
                                1E8D  ;--
                                1E8D
                                1E8D  EXIT_HANDLER:
                          0FFC  1E8D           .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                1E8F
                                1E8F           $SETSFM_S ENBFLG = #0           ; Turn off System Service failure mode
                                1E98           $SETAST_S ENBFLG = #0           ; An AST now could confuse us
                    00     EF   1EA1           EXTZV   #STS$V_SEVERITY,-       ; Save the proper exit severity...
                    03          1EA3                   #STS$S_SEVERITY,-
       50   0028'CF             1EA4                   EXIT_STATUS,R0
               03 50       E9   1EA8           BLBC    R0,10$                  ; ...as modified by the need to see...
            50     03      D0   1EAB           MOVL    #STS$K_INFO,R0          ; ...our message go into SYS$ERROR
                                1EAE  10$:
 50   00741080 8F          C8   1EAE           BISL2   #UETP$_ENDEDD,R0        ; ...and use it in our message code
      0004'CF      50      D0   1EB5           MOVL    R0,CLIG_ANNOUNCE+4
                                1EBA           $PUTMSG_S MSGVEC = CLIG_ANNOUNCE,- ; Output the ending message
                                1EBA                   ACTRTN = SE_COPY
                                1ECD           BBCW    #CLIG_V_SLAVE,FLAGS,40$ ; Skip this if we're the master proc
                                1ED6  ;
                                1ED6  ; Send our logged copy of SYS$ERROR to the master process.
                                1ED6  ;
                                1ED6           $REWIND RAB = SE_RAB            ; Set up to relay non-success msgs
      5A   0E02'CF         DE   1EE1           MOVAL   ERRORLOG_MSG,R10        ; Set up convenience registers...
      59   0E0C'CF         DE   1EE6           MOVAL   ERRORLOG_ENDED_MSG,R9   ; ...
0AA2'CF       02 AA    6A  28   1EEB           MOVC3   (R10),2(R10),MESSAGE_BUFFER ; Set up message preamble
   54       021A 8F    6A  A3   1EF2           SUBW3   (R10),#2*TEXTB_SIZE,R4  ; Figure length of buffer remaining
          1504'CF      53  D0   1EF8           MOVL    R3,SE_RAB+RAB$L_UBF     ; Set up RAB to automatically...
          1500'CF      54  B0   1EFD           MOVW    R4,SE_RAB+RAB$W_USZ     ; ...concatenate data with preamble
                                1F02  ;
```

```
                                     1F02  ; Send a dummy ERRORLOG message.  If messages are out of synch, this will
                                     1F02  ; cause the master to think it got a "garbled message", and the only messages
                                     1F02  ; it will attempt to read after that will be further ERRORLOG messages.  It
                                     1F02  ; also means that the first real ERRORLOG message will not be forgotten as
                                     1F02  ; a "garbled" message.  The master knows enough to ignore empty messages.
                                     1F02  ;
  63   54   00   00 8F   00   2C     1F02            MOVC5   #0,#0,#0,R4,(R3)        ; Clear out miscellaneous trash
                                     1F09  20$:
                         5A   DD     1F09            PUSHL   R10                     ; Define the type of message we want
              F8F2 CF    01   FB     1F0B            CALLS   #1,SLAVE_EXIT_WRITE     ; Pass a message to the master
                    33   50   E9     1F10            BLBC    R0,30$                  ; Exit loop if error
            00   00 8F   00   2C     1F13            MOVC5   #0,#0,#0,-              ; Clear out miscellaneous trash
                  1500'CF            1F18                    SE_RAB+RAB$W_USZ,-
                  1504'DF            1F1B                    @SE_RAB+RAB$L_UBF
                                     1F1E            $GET    RAB = SE_RAB            ; Get the next non-success message
                    DD   50   E8     1F29            BLBS    R0,20$                  ; Loop to write next msg if all is well
           50   00000000'8F   D1     1F2C            CMPL    #RMS$_EOF,R0            ; Have we finished copying?
                         11   13     1F33            BEQL    30$                     ; BR if so - send ending message
                  09CD'CF     28     1F35            MOVC3   PLEASE_CHECK_MSG,-      ; We have trouble with SYS$ERROR.LOG...
                  09D5'CF            1F39                    PLEASE_CHECK_MSG+8,-
                  1504'DF            1F3C                    @SE_RAB+RAB$L_UBF
                         5A   DD     1F3F            PUSHL   R10
              F8BC CF    01   FB     1F41            CALLS   #1,SLAVE_EXIT_WRITE     ; ...do our best to pass a warning
                                     1F46  30$:
            00   02 A9   69   2C     1F46            MOVC5   (R9),2(R9),#0,-         ; Insert our last message & clear rest
                  021A 8F            1F4B                    #2*TEXTB_SIZE,-
                  0AA2'CF            1F4E                    MESSAGE_BUFFER
                         59   DD     1F51            PUSHL   R9                      ; Send a line to say that we're done
              F8AA CF    01   FB     1F53            CALLS   #1,SLAVE_EXIT_WRITE
                                     1F58            $CLOSE  FAB = SE_FAB            ; Clean up after ourself
                                     1F63            $ERASE  FAB = SE_FAB            ; Clean up after ourself
                                     1F6E  40$:
                                     1F6E            $SETPRN_S PRCNAM = CURNAM_DESC  ; Reset our process name
                              04     1F79            RET                             ; That's all folks!
                                     1F7A
                                     1F7A            .END    UETCLIG00
```

```
                                                                C 13
UETCLIG00                          VAX/VMS UETP Cluster Integration Test     16-SEP-1984 00:19:09  VAX/VMS Macro V04-00         Page  84
Symbol table                                                                  6-SEP-1984 10:00:47  [UETPSY.SRC]UETCLIG00.MAR;1        (46)

$$.TAB                        = 000016D3 R      03     DEADLOCK_COUNT                  00000080 R      03
$$.TABEND                     = 00001717 R      03     DEADLOCK_LENGTH               = 00000008
$$.TMP                        = 00100000               DEADLOCK_LOCKID                 00000084 R      03
$$.TMP1                       = 00000001               DEADLOCK_MSG                    00000DDD R      02
$$.TMP2                       = 000000CF               DEADLOCK_MSG_TIME               00000088 R      03
$$.TMPX                       = 00000000 R      04     DEADLOCK_OFF_MSG                00000632 R      02
$$.TMPX1                      = 0000000D               DEADLOCK_OFF_PTR                00000CC6 R      02
$$T1                          = 00000000               DEADLOCK_VICTIMS                00000078 R      03
$$T2                          = 00000006               DEADLOCK_WAIT                   0000007C R      03
ABORTC_MSG_PTR                  00000C66 R      02     DEADLOCK_WAIT_MSG               00000660 R      02
ACCESS_LENGTH                 = 00000006               DEBUG_BUFFER                    00000FFB R      03
ACCESS_MSG                      00000DE7 R      02     DEBUG_DLOCK_VICTIM_MSG          00000B18 R      02
ANNOUNCE_US                     000001FD R      05     DEBUG_EXTEND_MSG                00000C23 R      02
ARG_COUNT                       00000038 R      03     DEBUG_FAO_BUF                   00000D96 R      02
BLANK_LINE                      000000BF R      02     DEBUG_FILE_MSG                  00000B60 R      02
BLANK_LINE_PTR                  00000CD6 R      02     DEBUG_INTRO_MSG                 00000A09 R      02
BLOCK                           000000D9 R      02     DEBUG_NOFILE_MSG                00000B7D R      02
BRK$C_DEVICE                  = 00000001               DEBUG_NOSHARE_MSG               00000BB4 R      02
BRK$M_CLUSTER                 = 00000800               DEBUG_PTR                       00000FF3 R      03
BRKTHRU_ERRORS                  00000282 R      02     DEBUG_QIO_MSG_PTR               00000CFA R      02
BRKTHRU_TIMOUT                = 0000003C               DEBUG_READ_MSG                  00000A79 R      02
BUFFER                          00000CC4 R      03     DEBUG_REQ_LOCK_MSG              00000AAC R      02
BUFFER_PTR                      00000CBC R      03     DEBUG_SHARE_MSG                 00000BEE R      02
CANCEL_MSG                      00000958 R      02     DEBUG_TAK_LOCK_MSG              00000AE4 R      02
CANCEL_MSG_PTR                  00000CC6 R      02     DEBUG_WRITE_MSG                 00000A47 R      02
CCASTHAND                       00001D7D R      05     DEV$V_CLU                       ********    X   05
CHECK_DEADLOCK                  000007BA R      05     DEV$V_TRM                       ********    X   05
CHECK_LOCKS                     000005A3 R      05     DEVCHAR                         0000003E R      03
CHF$L_SIGARGLST               = 00000004               DLOCK_ENQ                       000006F9 R      02
CHF$L_SIG_ARG1                = 00000008               DOTTEST                         000000E7 R      02
CHF$L_SIG_ARGS                = 00000000               DUMP                            00000058 R      02
CHF$L_SIG_NAME                = 00000004               DVI$_DEVCHAR                  = 00000002
CLIG_ANNOUNCE                   00000000 R      03     DVI$_DEVNAM                   = 00000020
CLIG_M_BEGINMSG               = 00000008               END_OF_TESTING                  0000022C R      02
CLIG_M_DEADNODE               = 00000002               ERRORLOG_ENDED_LENGTH         = 0000000E
CLIG_M_DEBUG                  = 00000001               ERRORLOG_ENDED_MSG              00000E0C R      02
CLIG_M_SE_DEAD                = 00000004               ERRORLOG_LENGTH               = 00000008
CLIG_M_SLAVE                  = 00000002               ERRORLOG_MSG                    00000E02 R      02
CLIG_V_BEGINMSG               = 00000003               ERRORLOG_PTR                    00000CE6 R      02
CLIG_V_DEADNODE               = 00000001               ERROR_COUNT                     00000034 R      03
CLIG_V_DEBUG                  = 00000000               ERROR_EXIT                      00001E00 R      05
CLIG_V_SE_DEAD                = 00000002               ERROR_SIGNAL                    00001DAD R      05
CLIG_V_SLAVE                  = 00000001               EXCLUDE_MSG                     00000999 R      02
CLSIODB_ARGS                    00000D62 R      02     EXIT_DESC                       00000014 R      03
CLSIODB_FAIL                    000002F3 R      02     EXIT_HANDLER                    00001E8D R      05
CLSIODB_SCREWEY                 0000032C R      02     EXIT_STATUS                     00000028 R      03
CLSPTR                          000000A2 R      03     FAB$B_BID                     = 00000000
CLU$GL_CLUB                     ********    X   05     FAB$B_DNS                     = 00000035
CLUSTER_MEMBER                  00000090 R      03     FAB$B_FAC                     = 00000016
COMMASPACE                      00000488 R      02     FAB$B_FNS                     = 00000034
COMMON_MSG                      00001B59 R      05     FAB$C_BID                     = 00000003
CONTINUE_LENGTH               = 00000008               FAB$C_BLN                     = 00000050
CONTINUE_MSG                    00000DEF R      02     FAB$C_SEQ                     = 00000000
CRLFTAB                         00000492 R      02     FAB$C_VAR                     = 00000002
CURNAM                          00000052 R      03     FAB$L_ALQ                     = 00000010
CURNAM_DESC                     0000004A R      03     FAB$L_DNA                     = 00000030
DC$_DISK                        ********    X   05     FAB$L_FNA                     = 0000002C
```

```
UETCLIG00                              VAX/VMS UETP Cluster Integration Test    16-SEP-1984 00:19:09  VAX/VMS Macro V04-00        Page  85
Symbol table                                                D 13                  6-SEP-1984 10:00:47  [UETPSY.SRC]UETCLIG00.MAR;1       (46)

FAB$L_FOP             = 00000004                 MOVE_ON_MSG             00000DF9  R    02
FAB$L_STS             = 00000008                 MYNODE_ITMLST           00000D26  R    02
FAB$L_STV             = 0000000C                 MYPROC_ITMLST           00000D52  R    02
FAB$M_PUT             = 00000001                 NAM$B_ESS             = 0000000A
FAB$V_CHAN_MODE       = 00000002                 NAM$B_NOP             = 00000008
FAB$V_FILE_MODE       = 00000004                 NAM$B_RSL             = 00000003
FAB$V_GET             = 00000001                 NAM$B_RSS             = 00000002
FAB$V_LNM_MODE        = 00000000                 NAM$C_BID             = 00000002
FAB$V_PUT             = 00000000                 NAM$C_BLN             = 00000060
FAB$V_SUP             = 00000002                 NAM$C_MAXRSS          = 000000FF
FAB$V_UPI             = 00000006                 NAM$L_ESA             = 0000000C
FAB$W_GBC             = 00000048                 NAM$L_RSA             = 00000004
FAO_BUF                 00000D8E  R    02        NEWNAM                  00000069  R    03
FILE                    0000011D  R    02        NEWNAM_DESC             00000061  R    03
FILE_ACCESS             00000DB2  R    05        NODE_CHANS              000000AA  R    03
FIVE_SECONDS            00000D86  R    02        NODE_LENGTH           = 00000006
FLAGS                   00000024  R    03        NODE_LIST_MSG           0000045B  R    02
GARBLED_TRANS           00001B47  R    05        NODE_LIST_MSG_PTR       00000CA6  R    02
GARBLE_MSG              00000918  R    02        NODE_NAMES              000002AA  R    03
GET_DEADLOCK            00000B97  R    05        NOT_MSG                 00000B54  R    02
GET_NODES               000002D2  R    05        NO_BLOCK_LOCK           00000583  R    02
GIVE_DEBUG_MSG          00001BA9  R    05        NO_DLOCK_SETUP          000005CB  R    02
GOTLOCK_LENGTH        = 00000007                 NO_DLOCK_SETUP_PTR      00000CB6  R    02
GOTLOCK_MSG             00000DC9  R    02        NO_FILE_NODE            000007E8  R    02
HELLO_LENGTH          = 00000005                 NO_FILE_NODE_PTR        00000CC6  R    02
HELLO_MSG               00000DB2  R    02        NO_LOCK_ENQ             00000545  R    02
IMOK_LENGTH           = 00000004                 NO_NODE_MSG             00000418  R    02
IMOK_MSG                00000DB9  R    02        NO_NODE_MSG_PTR         00000C96  R    02
INDENT                = 00000004                 NO_RMS_AST_TABLE        00000D9E  R    02
INPUT_ITMLST            00000D0A  R    02        NO_SLAVE_BLOCK          00000735  R    02
IO$M_CTRLCAST         = 00000100                 NRAT_LENGTH           = 00000014
IO$_READVBLK          = 00000031                 NULL                    000000BB  R    02
IO$_SETMODE           = 00000023                 OPA0                    00000064  R    02
IO$_WRITEVBLK         = 00000030                 OTHERNODE_ITMLST        00000D42  R    02
JPI$_PRCNAM           = 0000031C                 OTS$CVT_L_TI            ********    X  05
LCK$K_EXMODE          = 00000005                 PATTERN_1             = 0000005A
LCK$M_CONVERT         = 00000002                 PATTERN_2             = 000000F0
LCK$M_DEQALL          = 00000001                 PB$C_ENAB             = 00000002
LCK$M_NOQUEUE         = 00000004                 PB$C_OPEN             = 00000003
LIB$SIGNAL              ********    X  05        PB$S_STATE            = 00000002
LINK_FAILED             00000363  R    02        PB$V_STATE            = 00000001
LONELY_MSG              00000176  R    02        PLEASE_CHECK_MSG        000009CD  R    02
LONELY_MSG_PTR          00000C76  R    02        PRCNAM_LENGTH         = 0000000F
MASTER                  000000AD  R    02        PROCESS_NAME            00000000  R    02
MASTER_ERRORLOG_READ    00001A3E  R    05        QIO_DELTA               00000D76  R    02
MASTER_NODE             0000009C  R    03        QIO_TIMEOUT           = 0000003C
MASTER_NODE_DESC        00000094  R    03        QUAD_STATUS             0000002C  R    03
MASTER_READ             000019B0  R    05        QUEUELOCK_LENGTH      = 00000009
MASTER_WRITE            00001922  R    05        QUEUELOCK_MSG           00000DD2  R    02
MAX_MSGNAM_LENGTH     = 0000000E                 RAB$B_RAC             = 0000001E
MAX_NODES             = 000000FF                 RAB$C_BID             = 00000001
MEMB_PATH               00000782  R    02        RAB$C_BLN             = 00000044
MEMB_PATH_PTR           00000CC6  R    02        RAB$C_SEQ             = 00000000
MESSAGE_BUFFER          00000AA2  R    03        RAB$L_CTX             = 00000018
MESSAGE_NAMES           00000DB2  R    02        RAB$L_FAB             = 0000003C
MODE                    0000004C  R    02        RAB$L_RBF             = 00000028
MOVE_ON_LENGTH        = 00000007                 RAB$L_ROP             = 00000004
```

```
RAB$L_STS                       = 00000008                  STATUS_BUFFER                      00000EE6 R       03
RAB$L_STV                       = 0000000C                  STATUS_PTR                         00000EDE R       03
RAB$L_UBF                       = 00000024                  STATUS_STRING                      00000158 R       02
RAB$V_NLK                       = 00000014                  STATUS_TO_TEXT                     00001BC3 R       05
RAB$W_RSZ                       = 00000022                  STS$K_ERROR                      = 00000002
RAB$W_USZ                       = 00000020                  STS$K_INFO                       = 00000003
READ_FAILED                       00001B29 R       05       STS$K_SEVERE                     = 00000004
READ_MSG                          000008E0 R       02       STS$K_SUCCESS                    = 00000001
REBEL_MSG                         000001A9 R       02       STS$K_WARNING                    = 00000000
REBEL_MSG_PTR                     00000C86 R       02       STS$M_INHIB_MSG                  = 10000000
RECORD                            00000129 R       02       STS$S_FAC_NO                     = 0000000C
REPORT                            00000031 R       02       STS$S_SEVERITY                   = 00000003
RESULT_FILESPEC                   0000181E R       03       STS$V_FAC_NO                     = 00000010
RF_FAB                            00001623 R       03       STS$V_SEVERITY                   = 00000000
RF_FILESPEC                       0000171F R       03       SYI$_CLUSTER_MEMBER              = 000010CF
RF_FILESPEC_DESC                  00001717 R       03       SYI$_DEADLOCK_WAIT               = 0000105E
RF_NAM                            00001673 R       03       SYI$_SCSNODE                     = 00001067
RF_RAB                            000016D3 R       03       SYS$ASSIGN                         ********   GX    05
RMS$_BLN                          ********   X     02       SYS$BRKTHRUW                       ********   GX    05
RMS$_BUSY                         ********   X     02       SYS$CANCEL                         ********   GX    05
RMS$_CDA                          ********   X     02       SYS$CANTIM                         ********   GX    05
RMS$_DNF                          ********   X     05       SYS$CANWAK                         ********   GX    05
RMS$_EOF                          ********   X     05       SYS$CLOSE                          ********   GX    05
RMS$_FAB                          ********   X     02       SYS$CMKRNL                         ********   GX    05
RMS$_FACILITY                   = 00000001                  SYS$CONNECT                        ********   GX    05
RMS$_RAB                          ********   X     02       SYS$CREATE                         ********   GX    05
RMS_ERROR                         00001D10 R       05       SYS$DCLEXH                         ********   GX    05
RMS_ERR_STRING                    00000137 R       02       SYS$DEQ                            ********   GX    05
SCSNODE                           00000042 R       03       SYS$ENQ                            ********   GX    05
SET_UP_SLAVE                      00000541 R       05       SYS$ENQW                           ********   GX    05
SE_COPY                           00001CE0 R       05       SYS$ERASE                          ********   GX    05
SE_FAB                            00001430 R       03       SYS$EXIT                           ********   GX    05
SE_FILESPEC                       00001524 R       03       SYS$FAO                            ********    X    05
SE_NAM                            00001480 R       03       SYS$FAOL                           ********   GX    05
SE_RAB                            000014E0 R       03       SYS$FLUSH                          ********   GX    05
SHARE_ACCESS                      000012B2 R       05       SYS$GET                            ********   GX    05
SHORT                             0000003F R       02       SYS$GETDVIW                        ********   GX    05
SHR$_ABENDD                     = 000010E0                  SYS$GETJPI                         ********   GX    05
SHR$_BEGIND                     = 00001038                  SYS$GETMSG                         ********   GX    05
SHR$_ENDEDD                     = 00001080                  SYS$GETSYI                         ********   GX    05
SHR$_TEXT                       = 00001130                  SYS$GETSYIW                        ********   GX    05
SLAVE_EXIT_WRITE                  00001802 R       05       SYS$HIBER                          ********   GX    05
SLAVE_EXT_FAIL                    00000863 R       02       SYS$INPUT                          00000011 R       02
SLAVE_NO_ACCESS                   0000082A R       02       SYS$NET                            00000022 R       02
SLAVE_QIO_DELTA                   00000D7E R       02       SYS$OPEN                           ********   GX    05
SLAVE_READ                        000016D0 R       05       SYS$PUT                            ********   GX    05
SLAVE_WRITE                       00001769 R       05       SYS$PUTMSG                         ********   GX    05
SS$_CONTROLC                      ********   X     05       SYS$QIO                            ********   GX    05
SS$_DEADLOCK                      ********   X     05       SYS$QIOW                           ********   GX    05
SS$_NORMAL                        ********   X     05       SYS$REWIND                         ********   GX    05
SS$_NOTQUEUED                     ********   X     05       SYS$SCHDWK                         ********   GX    05
SS$_NOTRAN                        ********   X     05       SYS$SETAST                         ********   GX    05
SS$_SSFAIL                        ********   X     05       SYS$SETIMR                         ********   GX    05
SS$_WASSET                        ********   X     05       SYS$SETPRN                         ********   GX    05
SSERROR                           00001C15 R       05       SYS$SETSFM                         ********   GX    05
SS_SYNCH_EFN                    = 00000001                  SYS$TRNLOG                         ********   GX    05
START_TALKING                     000004D6 R       05       SYS$WAKE                           ********   GX    05
```

```
                                                    F 13
UETCLIG00                        VAX/VMS UETP Cluster Integration Test     16-SEP-1984 00:19:09  VAX/VMS Macro V04-00         Page  87
Symbol table                                                                6-SEP-1984 10:00:47  [UETPSY.SRC]UETCLIG00.MAR;1       (46)

SYS0_SYSTEST_DIR                    00000107 R      02
SYSTEST_DIR                         000000F6 R      02
TAKELOCK_LENGTH                   = 00000008
TAKELOCK_MSG                        00000DBF R      02
TAKE_OUT_LOCK                       000006D9 R      05
TASK                                00000071 R      02
TEXTB_SIZE                        = 0000010D
TIME_OUT                            00001AD9 R      05
TTCHAN                              0000003C R      03
UETCLIG                             0000009D R      02
UETCLIG00                           00000000 RG     05
UETP                              = 00740000
UETP$CLIG                           000000C7 R      02
UETP$CLSIODB                        ********    X   05
UETP$_ABENDD                      = 007410E0
UETP$_ABORTC                      = 0074832B
UETP$_BEGIND                      = 00741038
UETP$_COPY_LOG                    = 007480B1
UETP$_COPY_LOG_ENDED              = 007480C1
UETP$_COPY_LOG_LINE               = 007480B9
UETP$_DATADEVERR                  = 00748018
UETP$_ENDEDD                      = 00741080
UETP$_ERBOXPROC                   = 00748020
UETP$_FACILITY                    = 00000074
UETP$_TEXT                        = 00741130
UID$K_SID_RTYPE                   = 00000001
UIDDDB$A_FLINK                    = 00000000
UIDDDB$L_UCB                      = 00000007
UIDDDB$T_NAME                     = 0000000B
UIDFLAG$M_DDB                     = 00000004
UIDFLAG$M_MYSYS                   = 00000020
UIDFLAG$M_PATH                    = 00000002
UIDFLAG$M_SID                     = 00000001
UIDFLAG$M_UCB                     = 00000008
UIDGNRC$B_TYPE                    = 00000006
UIDPATH$B_RSTATE                  = 0000000D
UIDPATH$W_STATE                   = 00000007
UIDSID$A_FLINK                    = 00000000
UIDSID$L_DDB                      = 00000041
UIDSID$L_PBFL                     = 00000007
UIDSID$T_NODENAME                 = 00000031
UIDSID$T_SWTYPE                   = 00000011
UIDSID$T_SWVERS                   = 00000015
UIDUCB$A_FLINK                    = 00000000
UIDUCB$B_DEVCLASS                 = 00000009
UIDUCB$L_DEVCHAR2                 = 0000000F
UIDUCB$W_NUMBER                   = 00000007
UNIT_LENGTH                       = 00000005
VICTIMS_MSG                         000006B8 R      02
VMS                                 00000099 R      02
WARN_OF_TESTING                     000001D4 R      02
WIND_DOWN                           0000150D R      05
WRITE_FAILED                        00001B38 R      05
WRITE_MSG                           000008A9 R      02
WRONG_ENQ                           0000049D R      02
```

```
                                                 +----------------+
                                                 ! Psect synopsis !
                                                 +----------------+

PSECT name                      Allocation            PSECT No.  Attributes
----------                      ----------            ---------  ----------
.  ABS  .                       00000000  (     0.)   00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                           00000000  (     0.)   01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
RODATA                          00000E1C  (  3612.)   02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD   NOWRT NOVEC PAGE
RWDATA                          0000191D  (  6429.)   03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD     WRT NOVEC PAGE
$RMSNAM                         0000000D  (    13.)   04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
_UETP$CODE                      00001F7A  (  8058.)   05 (  5.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD   NOWRT NOVEC PAGE

                                              +------------------------+
                                              ! Performance indicators !
                                              +------------------------+

Phase                      Page faults    CPU Time       Elapsed Time
-----                      -----------    --------       ------------
Initialization                      29    00:00:00.09    00:00:00.85
Command processing                 153    00:00:00.79    00:00:04.09
Pass 1                             872    00:00:40.57    00:01:15.32
Symbol table sort                    0    00:00:03.36    00:00:06.42
Pass 2                             538    00:00:11.63    00:00:21.30
Symbol table output                  3    00:00:00.33    00:00:00.73
Psect synopsis output                3    00:00:00.03    00:00:00.03
Cross-reference output               0    00:00:00.00    00:00:00.00
Assembler run totals              1600    00:00:56.80    00:01:48.74

The working set limit was 2000 pages.
236763 bytes (463 pages) of virtual memory were used to buffer the intermediate code.
There were 120 pages of symbol table space allocated to hold 2079 non-local and 164 local symbols.
3485 source lines were read in Pass 1, producing 63 object records in Pass 2.
86 pages of virtual memory were used to define 78 macros.

                                             +--------------------------+
                                             ! Macro library statistics !
                                             +--------------------------+

Macro library name                           Macros defined
------------------                           --------------
_$255$DUA28:[SHRLIB]UETP.MLB;1                            2
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                            2
_$255$DUA28:[SYSLIB]STARLET.MLB;2                        63
TOTALS (all libraries)                                   67

2438 GETS were required to define 67 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:UETCLIG00/OBJ=OBJ$:UETCLIG00 MSRC$:UETCLIG00/UPDATE=(ENH$:UETCLIG00)+EXECML$/LIB+SHRLIB$:UETP/LIB
```

0426 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SATSUT13
LIS

SUCCOMMON
LIS

SATSUT07
LIS

SATSUT09
LIS

SATSUT11
LIS

UETDR7800
LIS

UETCLIG00
LIS

SATSUT14
LIS

SATSUT10
LIS

SATSUT08
LIS

SATSUT12
LIS